UNIVERSIDAD POLITÉCNICA DE PUEBLA

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

Programa Académico de Posgrado

Herramienta de cómputo en SIMULINK-MATLAB para el prototipado rápido y de bajo costo de Moduladores ΣΔ en FPAAs.

TESIS
QUE PARA OBTENER EL GRADO DE

MAESTRÍA EN INGENIERÍA EN SISTEMAS Y CÓMPUTO INTELIGENTE.

PRESENTA:
Yazmin Gaytán Luis.

DIRECTOR
Dr. Luis Abraham Sánchez Gaspariano

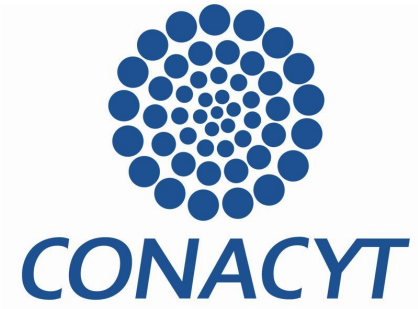
CO-DIRECTOR
Dr. Carlos Muñiz Montero

Posgrado UPPue

Juan C. Bonilla, Puebla, México, 2018.

El presente trabajo fue realizado en el Laboratorio de Investigación y Posgrado del Departamento de Sistemas y Computo Inteligente de la Universidad Politécnica de Puebla, ubicada en Tercer carril del Ejido "Serrano" S/N, San Mateo Cuanalá, Municipio Juan C. Bonilla, Puebla CP 72640.

Apoyo del CONACYT, Beca No. 701955, Programa de Maestría perteneciente al Programa Nacional de Posgrados de Calidad (PNPC-CONACYT). Apoyo del CONACYT a través del Proyecto Ciencia Básica 181201.



La presente tesis titulada **"Herramienta de cómputo en SIMULINK-MATLAB para el prototipado rápido y de bajo costo de Moduladores ΣΔ en FPAAs"** y realizada por la Ing. Yazmin Gaytán Luis, ha sido revisada y aprobada por el Jurado para obtener el Título de:

MAESTRA EN INGENIERÍA EN SISTEMAS Y COMPUTO INTELIGENTE

UNIVERSIDAD POLITÉCNICA DE PUEBLA

PNPC-CONACYT

Jurado integrado por:

Profesor	**Firma**
Director: Luis Abraham Sánchez Gaspariano	_____
Co-director: Dr. Carlos Muñiz Montero	_____
Revisor: Dr. Víctor Rodolfo González Díaz	_____
Revisor: Dr. Jesús Manuel Muñoz Pacheco	_____

Juan C. Bonilla, Puebla, México, a 10 de abril de 2018.

Agradecimientos

A:

Dios, *por fortalecer mi vida e iluminar mi mente y darme la oportunidad de concluir una meta más.*

Mi familia, *mis padres por forjarme como una persona perseverante y su constante interés en mi crecimiento profesional.*

Mi hermana, por el apoyo económico y moral incondicional que siempre ha tenido.

Mi hermano, que con el ejemplo de lucha, superación y amor por todo lo que hacía durante el tiempo que estuvo en este mundo, es suficiente para ser mi luz en la oscuridad.

Dr. Luis Abraham Sánchez Gaspariano y Dr. Carlos Muñiz Montero, *por darme la oportunidad de realizar el proyecto de tesis bajo su supervisión, además de la confianza, el tiempo, la paciencia y sobre todo sus conocimientos compartidos.*

Amigos, compañeros y profesores, *por brindarme su apoyo, conocimientos, experiencias, y momentos inolvidables.*

Tabla de contenido

Nomenclatura

A/D	Analog/Digital
ADC	Analog-to-digital converter
D/A	Digital/Analog
DAC	Digital-to-analog converter
DC	Direct current
DR	Dynamic Range
EDA	Electronic Desing Automation
ENOB	Effective number of bits
FPAA	Field Programmable Analog Array
FPGA	Field Programmable Gate Array
IC	Integrated circuit
IMD	Inter Modulation Distortion
NTF	Noise Transfer Funtion
ΣΔ	Sigma Delta
SC	Switched-capacitor
SFDR	Spurious-free dynamic range
SNR	Signal-to-noise ratio
STF	Signal transfer function
TC	Continuous Time
TD	Discrete time

Capítulo 1 Planteamiento del problema de investigación

1.1 Introducción

Durante varios años las técnicas de procesamiento analógico fueron ampliamente utilizadas para el acondicionamiento de señales eléctricas en equipo electrónico; sin embargo, con el auge de los circuitos integrados (CI), las técnicas de procesamiento digital poco a poco ganaron terreno, convirtiéndose en las más populares hoy en día para la casi totalidad de los sistemas electrónicos [1].

El procesamiento digital de señales requiere de los convertidores de datos, que a su vez están compuestos por Moduladores, quienes convierten una señal analógica en digital y viceversa [2]. Si bien, alrededor de un 90% de un sistema electrónico es digital, el 10% restante es analógico y/o mixto (i.e. con ambos tipos de señales, analógicas y digitales). Los sistemas mixtos consisten en los convertidores de datos analógico-digitales (ADCs por sus siglas en inglés) y los digital-analógicos (DACs por sus siglas en inglés). En función de la aplicación donde van a ser utilizados, existen diferentes tipos de convertidores de datos, entre los cuales se encuentran los convertidores tipo: *flash, folding, pipeline, SAR* y *ΣΔ*. Estos últimos, típicamente se emplean para tener un mayor número de bits a una razón de procesamiento relativamente baja [3], [4].

Por otra parte, las herramientas EDA (siglas en inglés de *Electronic Design Automation*) son utilizadas en el diseño de sistemas electrónicos tanto en *software* como en *hardware*, facilitándole al diseñador de circuitos la realización de sistemas eficientes, hoy en día los circuitos integrados (CIs) son cada vez más complejos por el gran número de componentes que los integran [1]. Es por esto que en este proyecto se utilizará de los diferentes tipos de herramientas que conforman EDA la herramienta SIMULINK/MATLAB, donde se desarrollarán los modelos

comportamentales de los diversos bloques que integran los convertidores sigma-delta (*ΣΔ*).

En el diseño de *hardware* existe el problema del alto costo y tiempo en el desarrollo del prototipo y pruebas para la validación de resultados. Para solucionar este problema, se realiza el proceso de simulación y verificación antes de la fabricación del circuito integrado (CI) o antes de bajarlo a alguna tarjeta programable (FPGAs, FPAAs &/o PSOCs). Si los resultados son los esperados se comienza la implementación del prototipo. Hasta aquí, las herramientas EDA se utilizan en cada una de las fases del diseño. Una de las tarjetas programables utilizadas por los diseñadores son las FPAAs (siglas en inglés de *Field programmable analog arrays*) que tienen muchas ventajas tales como la disminución del tiempo para el desarrollo de su tarea por la cual fue programada, reduce los costos de ingeniería tales como la investigación, el diseño y las pruebas de un nuevo producto. FPAA no sólo es óptimo para toda la solución en contraste con FPGAs sino que también reduce el tiempo de verificación y costo de diseños. Esto de nuevo resulta de la naturaleza compleja de los circuitos analógicos que necesita factores como la relación señal-ruido, ancho de banda, Respuesta de frecuencia, linealidad entre otras [5].

La velocidad en el crecimiento y desarrollo en la tecnología de CI ha conllevado nuevos retos y tareas complejas, a este respecto existen herramientas de simulación tales como Agilent Advanced Design System, Hspice, SIMULINK, entre otras, que han ayudado a enfrentar dichos retos. Como consecuencia, esta investigación tiene como propósito implementar una herramienta que nos permita el prototipado rápido y de bajo costo de convertidores de datos analógicos a digitales del tipo sigma-delta (ΣΔ), todo esto implementado en SIMULINK/MATLAB, donde finalmente los resultados se trasladarán mediante una interfaz con una tarjeta de adquisición de datos a *hardware* reconfigurable FPAA.

1.2 Objetivos

1.2.1 Objetivo general

Implementar una herramienta en SIMULINK/MATLAB que permita el prototipado en corto tiempo y bajo costo del Modulador de datos analógico-digital del tipo ΣΔ.

1.2.2 Objetivos específicos

- Realizar modelos comportamentales en SIMULINK/MATLAB de los bloques que integra la arquitectura del Modulador ΣΔ.
- Verificar el correcto funcionamiento de cada uno de los bloques realizados mediante simulaciones en MATLAB.
- Trasladar sistemas dinámicos modelados en SIMULINK/MATLAB a hardware reconfigurable (FPAA) mediante una interface y validar experimentalmente.

1.3 Justificación

En la actualidad la fabricación de un Circuito Integrado es muy tardado y con un precio muy elevado; de acuerdo a los datos proporcionados por *EUROPRACTICE* fabricante de CI (precios actualizados del año 2016) [6] un CI oscila entre los cientos de miles de Dlls dependiendo del área del mismo, con el inconveniente de que el tiempo de entrega es de 6 meses en adelante de acuerdo a la complejidad del diseño [7]. El costo del prototipo que se propone en este trabajo es mucho más bajo, aproximadamente de 1,514 Dlls, considerando el costo de la tarjeta FPAA Anadigm AN231E04 [5]. Con la ventaja de que con la implementación de los modelos de los convertidores del tipo sigma-delta que se desarrollarán en SIMULINK/MATLAB, la implementación sería casi inmediata. Con base a los datos mostrados anteriormente podemos decir que el proyecto reduce significativamente costos y tiempo.

Además, el diseño de los ADCs ΣΔ en SIMULINK/MATLAB se realizarán por bloques, esto les facilita a investigadores y desarrolladores diseñar otros circuitos electrónicos tales como osciladores, filtros y moduladores, por citar algunos.

1.4 Organización de Tesis

La organización del documento de tesis es la siguiente: en el capítulo 1 se describe el planteamiento del problema además de los objetivos específicos y general, así como la justificación; por otra parte, en el capítulo 2 se da una descripción de los temas relacionados al proyecto; posteriormente en el capítulo 3 se menciona la metodología a utilizar para el desarrollo del prototipo; así mismo en el capítulo 4 se muestran los resultados obtenidos tanto de las simulaciones como de del prototipado; y finalmente en el capítulo 5 las conclusiones.

Capítulo 2 Marco teórico

2.1 Antecedentes

La entrada de las computadoras (*hardware*) en el siglo XX basadas en los circuitos integrados (CI), genera lo denominado la “crisis del *software*”, en el año de 1968 la ingeniería del *software* comienza a tomar verdadero peso en nuevas técnicas y métodos para satisfacer la complejidad de grandes sistemas. La Ingeniería de *Software* que se deriva de las ciencias de la computación, se define como: “una disciplina de ingeniería que comprende todos los aspectos de la producción de *software* desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza” [8].

En el año 2010 Pressman [9] define a la Ingeniería de Software como una tecnología con varias capas como se muestra en la Figura 2.1, donde cualquier enfoque de ingeniería incluyendo software debe basarse en un compromiso organizacional de calidad.

Figura 2.1 Capas de la ingeniería de software [9].

Por lo que se refiere a los *métodos* de la Ingeniería de Software (IS), ellos proporcionan la experiencia técnica para elaborar programas. Incluyen un conjunto amplio de tareas, como comunicación, análisis de los requerimientos, modelación del diseño, construcción del programa, pruebas y apoyo. Los métodos de IS se basan en un conjunto de principios fundamentales que

gobiernan cada área de la tecnología e incluyen actividades de modelación. Las *herramientas* de IS proporcionan un apoyo automatizado para el proceso y los métodos. Cuando se integran las herramientas de modo que la información creada por una pueda ser utilizada por otra, queda establecido un sistema llamado ***ingeniería de software asistido por computadora*** que apoya el desarrollo de software.

La ingeniería de software asistida por computadora o mejor conocida por su acrónimo en inglés (CASE) proporciona ayuda al proceso de software automatizando en algunas de sus actividades como:

- El desarrollo de modelos gráficos del sistema como parte de la especificación de requerimientos o del diseño de software.
- La comprensión del diseño utilizando un diccionario de datos que tiene información sobre las entidades y relaciones del diseño.
- La generación de interfaces de usuario a partir de la descripción gráfica de la interfaz que es elaborada de forma interactiva por el usuario.

La *herramienta de construcción de prototipos y la herramienta de apoyo a métodos* son parte de la clasificación de las herramientas CASE acorde con su función, la primera es un lenguaje de alto nivel y generadores de interfaz de usuario, la segunda herramienta es un editor de diseño, diccionarios de datos y generadores de código, por mencionar algunos ejemplos. Actualmente existen diversas herramientas de apoyo tales como EDA, CAD, FMCAD, por mencionar algunas, estas herramientas han sido de gran aportación al desarrollo y crecimiento de la tecnología (circuitos integrados) en la industria desde la ingeniería química a bioinformáticas, desde diseño del automóvil hasta sistemas inteligentes de transporte, desde la ingeniería aeroespacial a nano-ingeniería. Por otra parte la herramienta EDA por su acrónimo en inglés “Electronic Design Automation” es un proceso que mejora la fabricación, el desarrollo y diseño de los productos (placas electrónicas, circuitos integrados, etc.) con la ayuda de la computadora. En relación con los circuitos integrados (CI), sirven como la base de la era de la información, la mejora de nuestra vida en una serie de formas: computadoras rápidas, teléfonos celulares, televisores, cámaras digitales, etc. En particular no es posible la fabricación de un CI sin necesidad de alguna herramienta y/o técnica que nos facilite su diseño. En las herramientas EDA podemos encontrar software que pertenecen a este grupo: Cadence Design Framework II, Mentor Graphics, Xilinx, Agilent Advanced Design System, Matlab/Simulink entre otras [8]. De donde SIMULINK/MATLAB destaca por ser *“una herramienta para simular sistemas dinámicos con una interfaz gráfica especialmente desarrollada para este propósito”* [10]. Razón por la cual en electrónica actualmente se diseñan convertidores analógico-digitales (A/D) y digital-analógico (D/A) en SIMULINK/MATLAB.

Los Moduladores han llegado a ser una parte indispensable en el procesamiento digital de señales para los sistemas electrónicos. Este bloque exige que los pasos se realicen sea de forma óptima para no perder información. Según el tipo de componente y su aplicación existen distintos

parámetros que lo caracterizan, éstos pueden ser: la velocidad de conversión, la resolución, los rangos de entrada, etc. Por ejemplo, una mayor cantidad de bits, implica mayor precisión, pero también mayor complejidad. Un incremento en un solo bit permite disponer del doble de precisión (mayor resolución), pero hace más difícil el diseño del circuito, además, la conversión podría volverse más lenta [2].

En el mercado existen diferentes tipos de Moduladores A/D donde los más sobresalientes son: SAR, folding, pipeline, flash y *ΣΔ*. La elección de la herramienta EDA en un diseño electrónico dependerá de sus características a los requerimientos de la aplicación. En el caso de los convertidores A/D *ΣΔ* se utilizan cuando se requiere resolución, este es un sistema no lineal (debido al efecto de cuantificación[1] del ADC), así como dinámica (debido a la memoria en el integrador), y por lo tanto su análisis es una tarea matemática difícil [11].

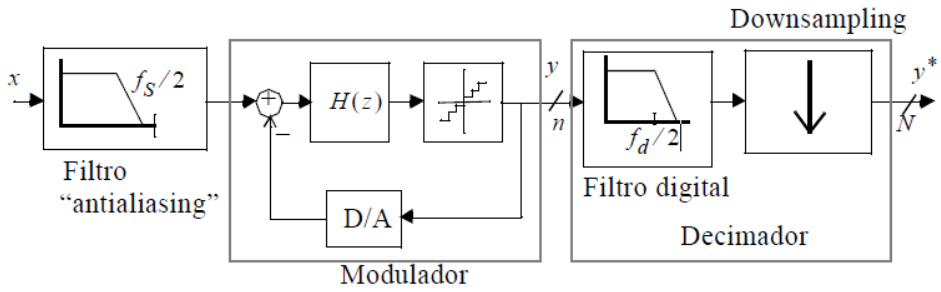


Figura 2.2 Diagrama de bloques de un convertidor A/D ΣΔ [12].

2.2 Convertidores A/D ΣΔ

La principal característica de un convertidor ΣΔ es la obtención de una señal digital de gran resolución, a una razón de procesamiento relativamente baja.

Un convertidor sigma-delta (ΣΔ) consta de tres bloques principales: Un filtro analógico o "antialiasing", un modulador ΣΔ y un decimador [13], como se muestra en la Figura 2.2.

[1] En la cuantificación digital el proceso de cuantificación es posterior a la etapa de muestreo en la que se toman valores de amplitud de señal analógica.

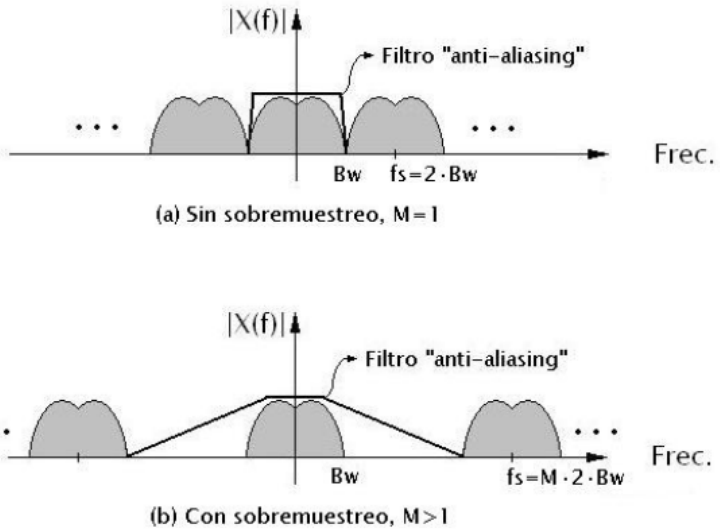


Figura. 2.3 Suavización de las especificaciones sobre el filtro anti-aliasing debido al sobremuestreo [14].

2.2.1 Filtro *Antialiasing*

Filtro analógico o "antialiasing": Es el que filtra la señal de interés Figura 2.3 y elimina las frecuencias superiores por encima de la mitad de la frecuencia de muestreo y así poder cumplir con el criterio de Nyquist (1), además de reducir la potencia del ruido que el transductor[2] de entrada agrega.

$$f_s \geq 2^* f_m \qquad (1)$$

La señal que entrega el filtro es conocida como «señal banda base», es decir, son señales que son transmitidas en su frecuencia original.

2.2.2 Modulador ΣΔ.

Modulador: En este bloque se realizan dos técnicas principales, denominadas muestreo y cuantización, que se explican a continuación:

[2] El transductor es un dispositivo que transforma una forma de energía en otra, por ejemplo, en la conversión de sonido a energía eléctrica el transductor es el micrófono.

Muestreo: Se encarga de tomar diferentes muestras de tensiones o voltajes (datos de la señal) de diferentes puntos de la onda de la señal de entrada y retenerlos durante un periodo de tiempo. En este proceso se realiza el sobre muestreo, que permite alcanzar una determinada calidad de señal sin necesidad de una tolerancia muy precisa de los componentes. Consiste en el incremento a la tasa de muestreo (La frecuencia a la que se realiza el muestreo) y es mucho mayor que la frecuencia de Nyquist (1), dando como resultado que las amplitudes de las dos muestras consecutivas sean pequeñas y así poder ser representadas en forma digital, usando muy pocos bits, normalmente de uno, la relación que existe entre la frecuencia de sobre muestreo y la frecuencia de Nyquist es [12]:

$$\mathrm{OSR}=\frac{f_B}{F_{Nyquist}} \tag{2}$$

Donde OSR es el factor de sobre muestreo y f_B es el ancho de banda de la señal análoga. Durante el proceso de muestreo se asignan valores numéricos equivalentes a la tensión o voltaje existente en diferentes puntos de la onda de la señal, con la finalidad de realizar a continuación el proceso de cuantización.

Cuantización: Es un proceso fundamental en los convertidores A/D, donde se convierten valores continuos, en series de valores discretos [15]. Esto quiere decir que las muestras que conforman la señal discreta se van a aproximar con un conjunto finito y acotado de niveles de amplitud. Sin embargo, en este proceso existe un error propio de la cuantización denominado “noise shaping” que para solucionarlo es necesario remover el ruido de la banda de interés y desplazarlo a frecuencias mayores mediante una función de transferencia con propiedades similares a las de un filtro pasa altas. Este comportamiento se puede obtener mediante la implementación de un sistema realimentado, que actúa como un filtro pasa-altas para la señal de entrada, tal y como el que se muestra en la etapa del modulador (Figura 2.2). Por otra parte, en la Figura 2.4 (a) muestra la característica de transferencia de un cuantizador de varios bits. Ésta puede representarse matemáticamente mediante una función no lineal de la forma [16]:

$$\mathrm{y}= g_q \mathrm{i} + e \tag{3}$$

Donde g_q es la pendiente de la recta que intersecta los pasos del código, o ganancia del cuantizador; y *e* representa el error de *cuantización*. Nótese en la Figura 2.4 (a) que, si la entrada del mismo permanece acotada en el intervalo [i_{min},i_{max}], el error de *cuantización* está acotado en el intervalo [-Δ/2, Δ/2], siendo Δ la separación entre niveles consecutivos del cuantizador. Esto también presenta una densidad espectral de potencias constantes como la del ruido blanco. Por esta razón se habla del error de *cuantización* como ruido de *cuantización*. Para entradas fuera del intervalo [i_{min},i_{max}], el valor absoluto del error de *cuantización* crece sin límite. Esta situación se conoce como sobrecarga del cuantizador.

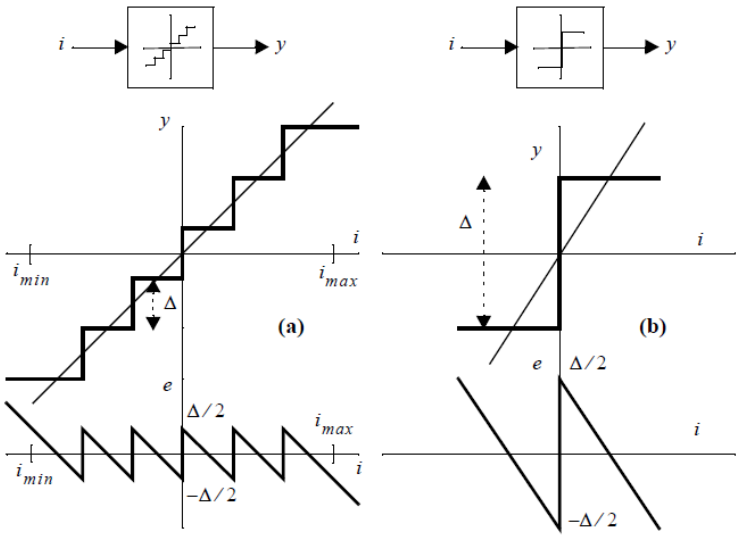


Figura 2.4 Características de transferencia y error de cuantización de [16]: (a) un cuantizador de varios bits; (b) un cuantizador de un bit (comparador)

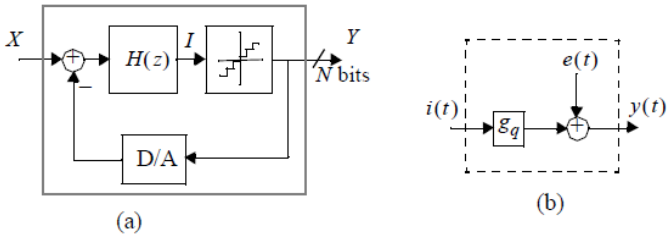


Figura 2.5 (a) Estructura básica del modulador ΣΔ. (b) Modelo del cuantizador.

Así pues, consideraremos para el cuantizador el modelo de la Figura. 2.5 (b) donde *e(t)* presenta una distribución uniforme entre sus valores máximo Δ/2 y mínimo - Δ/2 de forma que la densidad espectral de potencia del ruido de cuantización es constante.

Nótese que la ganancia que precede a la operación de cuantización en la Figura 2.4 (b) solo tiene sentido cuando el número de niveles del cuantizador sea superior a dos. El modulador de la Figura 2.5 (a) se convierte en un sistema de dos entradas *x(t)* y *e(t)*, y una salida *y(t)* que, en el dominio *Z,* puede representarse por [16]:

$$Y(z) = STF(z)X(z) + NTF(z)E(z) \tag{4}$$

Donde *X(z)* y *E(z)* son las transformadas Z de la señal de entrada y del ruido de cuantización respectivamente, y *STF*[3]*(z)* y *NTF*[4]*(z)* son las funciones de transferencia respectivas de la entrada y el ruido de cuantización. La forma exacta de ambas funciones depende de la arquitectura del modulador. Podemos imponer las condiciones siguientes para obtener moduladores operativos:

$$\mathrm{STF}(z) \rightarrow \mathrm{cte}$$

$$\mathrm{NTF}(z) \rightarrow 0 \qquad \text{para } z \rightarrow 1$$

Esto es, el ruido de cuantización debe atenuarse en la zona de bajas frecuencias sin que se distorsione la señal. La señal diferenciada, la cual es la parte "delta", es acumulada en el integrador, el cual es la parte "sigma" y provee un promedio local de la entrada.

El término del convertidor "sigma-delta" viene del hecho de que hay un punto de suma (sigma) y un modulador delta (integrador y cuantizador de 1 bit).

2.2.3 Decimador

Decimador: La función de este bloque consiste en la reducción de la frecuencia de muestreo. Consta de dos etapas: filtrado digital y *downsampling* (muestreo hacia abajo) como se muestra en la Figura 2.2. Otra forma gráfica de representarlo es tal como se muestra en la Figura 2.6, donde se toma cada *D* muestras equidistantes de la señal original, y se descartan las demás, dejando así una señal equivalente, pero con diferente tasa de muestreo. Como resultado se obtiene la señal de entrada, codificada con un elevado número de bits, a la frecuencia de Nyquist.

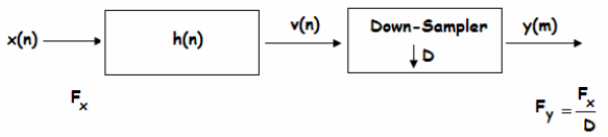


Figura 2.6 Decimación por un factor *D* [15].

[3] Por su sigla en inglés "Signal Transfer Funtion".
[4] Por su sigla en inglés "Noise Transfer Funtion".

La Figura 2.7 ilustra el procesado de la señal pasando por los distintos bloques del convertidor.

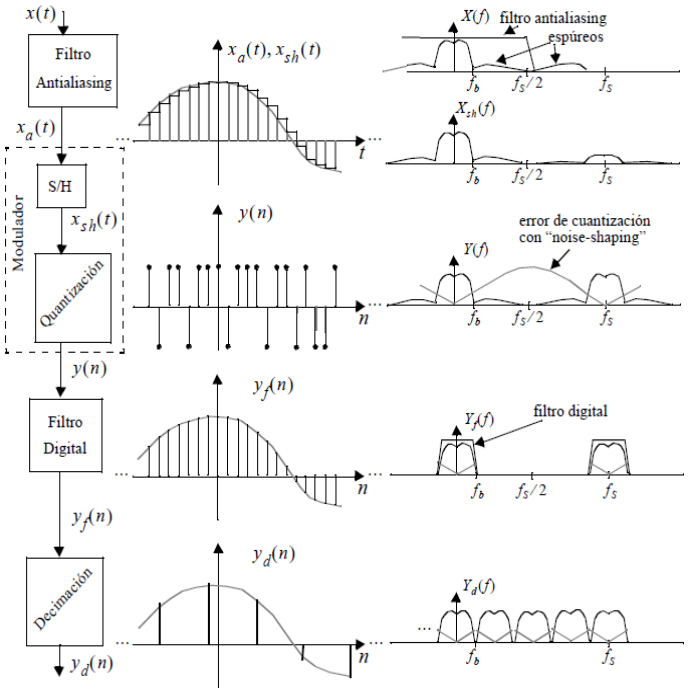


Figura 2.7 Ilustración del procesado de señal en un convertidor A/D ΣΔ [15].

2.2.4 Clasificación de las arquitecturas de moduladores ΣΔ.

La Figura 2.8 muestra una clasificación de las arquitecturas de moduladores ΣΔ describiendo las ventajas e inconvenientes y en la Tabla 2.1 se resume las diferencias entre los moduladores ΣΔ continuos y discretos.

Para mejorar el Rango Dinámico[5] de un modulador ΣΔ puede ser realizado de muchas maneras diferentes, dando lugar a la enorme cantidad de arquitecturas de moduladores. De acuerdo con la literatura, los moduladores ΣΔ se pueden agrupar atendiendo diferentes criterios de clasificación, por ejemplo [16]:

[5] Rango Dinámico se define como el cociente entre la potencia a la salida a la frecuencia de una sinusoide con amplitud igual al rango completo y la potencia a la salida para una sinusoide de la misma frecuencia y amplitud tal que resulte indistinguible del ruido; esto es, con SNR = 0dB. Usualmente se expresa en dB.

- Lazo único Vs Cascada (se enfocan al número de cuantificadores empleados). Empleando solo un cuantificador son llamados arquitecturas de lazo único, mientras las que emplean varios cuantificadores son llamados en cascada o MASH.
- Single-bit Vs Multi-bit (enfocados a número de bits en el cuantificador). Las arquitecturas de un solo bit o conocida como monobit son muy comunes ya que son menos propensos a los no ideales del circuito.
- Paso-bajas Vs Pasa-banda (se enfoca a la señal natural convertida). Los Moduladores Pasa-bajas consumen menos energía por paso de conversión a diferencia de los convertidores pasa-banda.
- Tiempo-Discreto Vs Tiempo-Continuo (atendiendo el lazo natural de los filtros dinámicos). Se diferencian en la operación de muestreo donde se trata la señal analógica/mixta. Donde un conmutador es el que simboliza el lugar donde ocurre el muestreo y la transición del tiempo continuo (TC) al tiempo discreto (TD).

Estas arquitecturas de los moduladores ΣΔ se verán nuevamente en el siguiente capítulo, donde podremos ver más a detalle todos los bloques que los componen, y así poder obtener las especificaciones de cada uno.

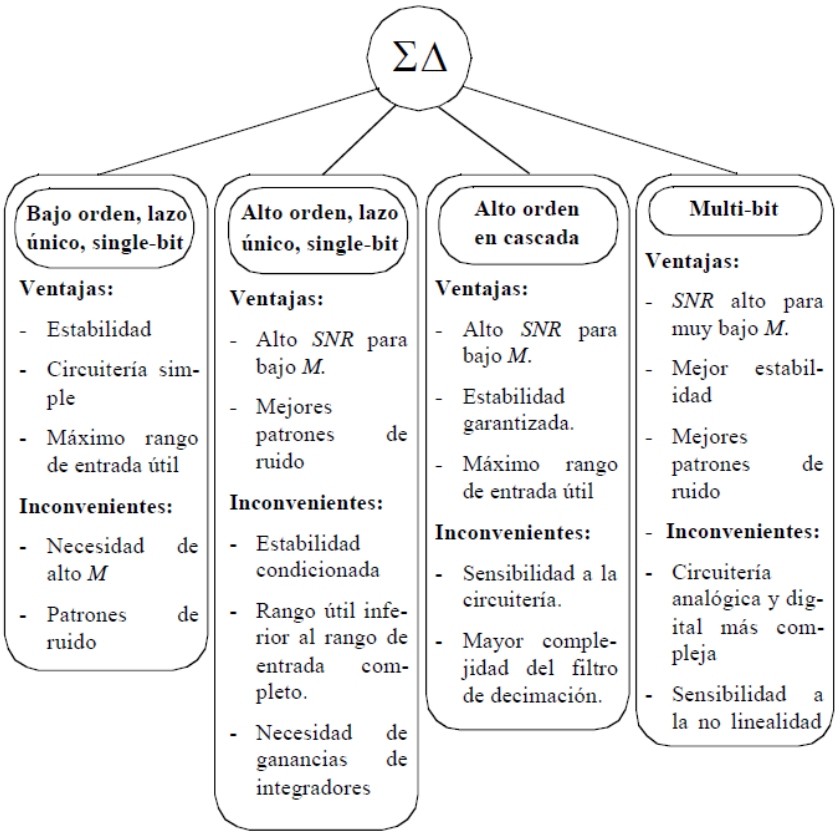


Figura 2.8 Clasificación de las arquitecturas de moduladores ΣΔ [16].

Tabla 2.1 Diferencias entre moduladores ΣΔ discretos y continuos [14].

Tiempo continuo	Tiempo discreto
⋆ Filtro *anti-aliasing* implícito	⋆ Baja sensibilidad al *clock jitter*
⋆ Menor error en la etapa de muestreo	⋆ Baja sensibilidad a los retardos en el lazo de realimentación
⋆ Mayores frecuencias de muestreo	⋆ Baja sensibilidad a los errores del *DAC*
⋆ Menores requerimientos de velocidad en sus componentes	⋆ Ganancias de cada integrador definidas precisamente
⋆ Reducción del efecto del ruido de la fuente de alimentación	⋆ Circuitos altamente lineales
⋆ Menor tiempo de simulación a nivel de circuito	⋆ Menor tiempo de simulación a nivel de sistema
	⋆ Únicamente cargas capacitivas
	⋆ Compatible con procesos CMOS VLSI

2.3 Métricas de desempeño

2.3.1 Resolución.

La resolución, ***q***, es la porción más pequeña de señal en la entrada del convertidor A/D que produce un cambio en el código digital a la salida del mismo. Matemáticamente se define como [17]:

$$1LSB=q=\frac{V_{FS}}{2^n} \quad (5)$$

Donde 1LSB indica un cambio del bit menos significativo a la salida del convertidor; n= número de bits que tiene el convertidor, V_{FS} = la excursión máxima de voltaje a la entrada del convertidor.

En la Figura 2.9 se muestra la relación entrada-salida de forma ideal de un convertidor A/D de 3 bits, la cual está dada por una característica escalonada, donde se toman en cuenta los puntos de decisión que se encuentran en el centro de cada intervalo de cuantización (1/2 LSB). La cuantización es el proceso de convertir valores continuos, en series de valores discretos, esto quiere decir que las muestras que conforman la señal discreta se van a aproximar con un conjunto finito y acotado de niveles de amplitud.

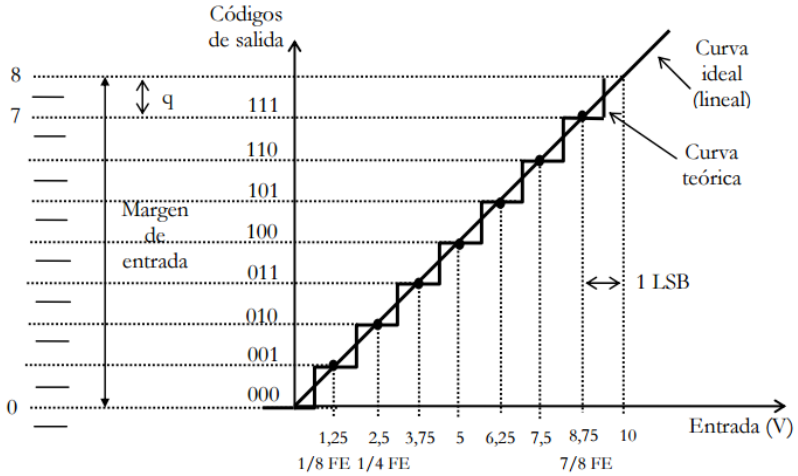


Figura 2.9 Curva de transferencia de un convertidor A/D de tres bits con cuantificación uniforme [14].

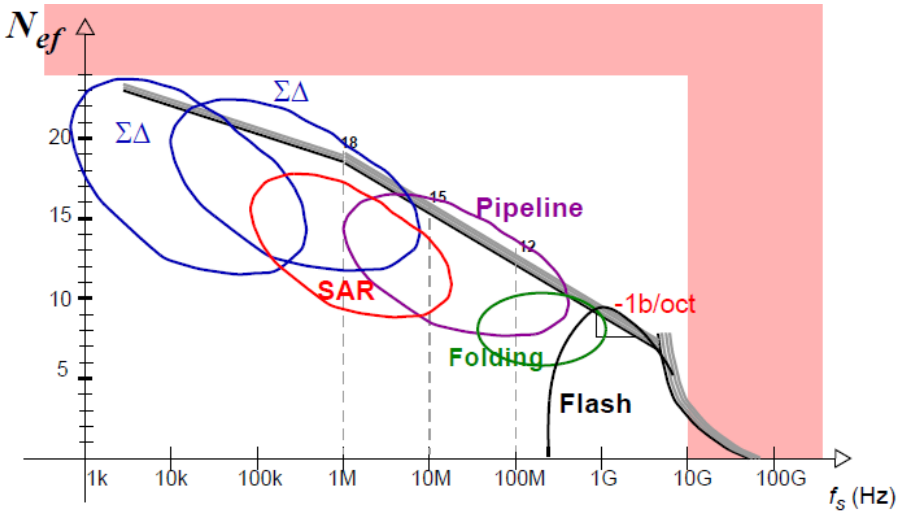


Figura 2.10 Arquitecturas de convertidores A/D de acuerdo a la velocidad y resolución [18].

En la práctica, el éxito de la conversión depende de las limitaciones de la implementación. De éstas, las más comunes son *la velocidad* (frecuencia de muestreo) y la *resolución* (número de bits). La elección del convertidor A/D en un diseño electrónico dependerá de los requerimientos de la aplicación.

Existen varios tipos de convertidores A/D, los que más sobresalen son:

- Aproximaciones sucesivas (successive approximation register, SAR).
- Pipeline.
- Sigma-Delta (delta-sigma).
- Flash.
- Foldeo.

En la Figura 2.10 se muestra una gráfica donde éstos se ubican de acuerdo a la frecuencia de muestreo y el número de bits (resolución) que manejan.

Los convertidores de alta velocidad tienen baja resolución y viceversa, el que más se sitúa con un buen compromiso entre resolución y velocidad es el convertidor SAR, el cual es muy popular en diversos sistemas electrónicos. En caso de requerir una velocidad alta, lo recomendable es utilizar una arquitectura Flash. Por lo contrario, si la resolución es importante entonces el convertidor que ofrece mejores prestaciones en este rubro es el ΣΔ. Las aplicaciones del convertidor ΣΔ en sistemas electrónicos son amplias y abarcan una buena parte de sistemas de instrumentación de alta precisión como es el caso de aquellos utilizados en telemetría.

2.3.2 Frecuencia de muestreo

El proceso de muestreo se encarga de tomar diferentes muestras de tensiones o voltajes (datos de la señal) de diferentes puntos de la onda de la señal de entrada y retenerlos durante un periodo de tiempo. Al periodo de tiempo que existe entre dos muestras consecutivas se le conoce con el nombre de periodo de muestreo (Ts). A partir del periodo de muestreo se calcula la frecuencia de muestreo (fs = 1/Ts), esta limita el ancho de banda (BW) de la señal de entrada. Al muestrear una señal nos podemos encontrar con los siguientes casos [18]:

- Una frecuencia de muestreo, fs, de baja velocidad (o un periodo de muestreo grande), lo que implica una menor cantidad de muestras por unidad de tiempo, fs < 1 Ms/s.
- Una frecuencia de muestreo, fs, de media-alta velocidad (o un periodo de muestreo medio), lo que implica una cantidad media-alta de muestras por unidad de tiempo, 1 Ms/s ≤ fs ≤100 Ms/s.
- Una frecuencia de muestreo, fs, de muy alta velocidad (o un periodo de muestreo muy pequeño), lo que implica una cantidad muy alta de muestras por unidad de tiempo, fs ≥ 100 Ms/s.

La idea de muestrear a una velocidad mayor que la necesaria se basa en que es posible intercambiar posteriormente velocidad (muestras/seg) por resolución (número de bits). Además de que un mayor número de muestras es la mejor garantía que podemos encontrar para poder recuperar la señal original a partir de las muestras. Sin embargo, tener una frecuencia de muestreo alta puede generar un volumen de información demasiado grande para nuestros intereses (demasiadas muestras a tratar).

El Teorema de Nyquist, (1), nos indica la frecuencia de muestreo mínima que hemos de utilizar para poder recuperar la señal original. Si se cumple el criterio de Nyquist en la fase de muestreo no se introduce ningún error.

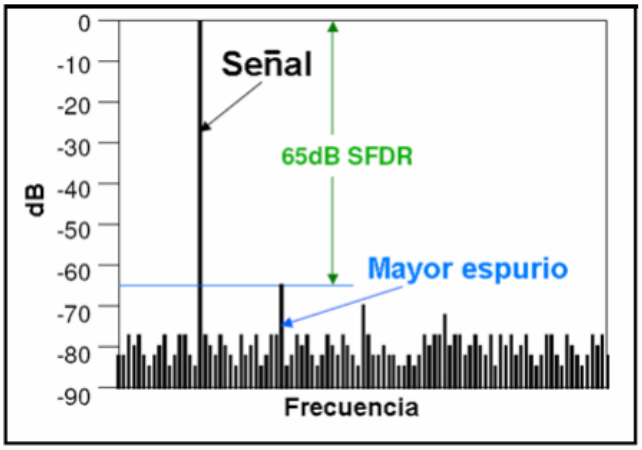


Figura 2.11 Gráfica del espectro de una señal con un SFDR de 65dB [18].

2.3.3 SNR y SFDR

La relación señal-ruido de cuantización SNR (Signal to Noise Ratio) se define como el cociente de varianzas de la señal y el ruido presente en el circuito. En el caso de un convertidor ideal el único ruido presente es el de cuantización y por tanto la SNR se define como [18]:

$$SNR = 10 \log \frac{P_{señal}}{P_{ruido\_cuantización}}$$

Expresado en dB:

$$SNR = 6.02N + 1.76 \quad dB \tag{6}$$

Donde N es el número de bits de conversión. Esto significa que por cada bit añadido a la palabra digital, la *SNR* aumenta aproximadamente 6 decibelios. Mientras mayor sea el número de bits del convertidor, menor será el ruido de cuantificación y mayor la *SNR*.

Rango dinámico libre de espurios (*SFDR*, *Spurious free dynamic range*) es la diferencia, expresada en dB, entre el valor *rms*[6] (en inglés, *root mean*

[6] El valor rms es el valor del voltaje o corriente en C.A. que produce el mismo efecto de disipación de calor que su equivalente de voltaje o corriente directa.

square) de la señal de entrada a la salida del convertidor y el valor rms del mayor espurio. La Figura 2.11 ilustra tanto la *SNR* como la *SFDR*. Un espurio se define como cualquier señal presente en el espectro de salida no incluido en la señal de entrada al convertidor [19]. Matemáticamente la SFDR se expresa como:

$$\text{SFDR}=10\log\frac{P_{señal}}{P_{espurio\_max}} \tag{7}$$

2.3.4 Distorsión por intermodulación (IMD)

Cuando dos o más señales se mezclan en un circuito, se produce la distorsión de intermodulación o *IMD* (*Inter Modulation Distortion*). La *IMD* no resulta agradable al oído en un sistema de audio, precisamente porque las frecuencias que genera no tienen una relación armónica con la señal original. En la Figura 2.12 se muestra la simulación del espectro de una señal formada por dos tonos *IMD* que se mide mediante la aplicación de dos ondas sinusoidales espectralmente puras para el convertidor A/D con las frecuencias f_1=5MHz y f_2=6MHz, cabe mencionar que f_1 y f_2 por lo general se encuentran muy próximas entre sí. La amplitud de cada tono se establece ligeramente más de 6 dB por debajo de la escala completa de manera que el convertidor no recorta cuando los dos tonos se suman en fase. Los productos de tercer orden $2f_2 - f_1$ y $2f_1 - f_2$ están cerca de las señales originales y son más difíciles de filtrar [19]. Matemáticamente *IMD* se define como:

$$IMD=10\log\frac{P_{señal}}{P_{espu\_IM}} \tag{8}$$

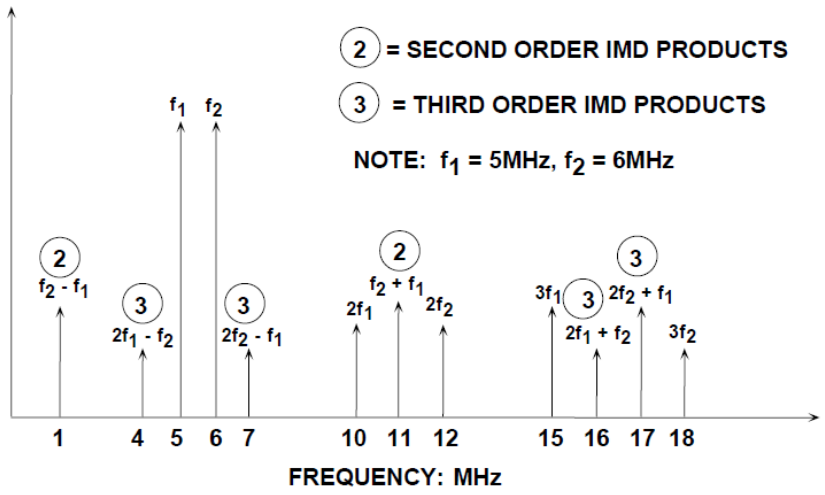


Figura 2.12 Productos de intermodulación de segundo y tercer orden para f_1 = 5 MHz y f_2 = 6 MHz [15]

2.3.5 ENOB

El número efectivo de bits *ENOB* (*Effective Number of Bits*) es una de las especificaciones dinámicas más utilizadas para convertidores A/D reales, ya que es un indicador global de la resolución del sistema para una determinada frecuencia de la señal de entrada a una cierta velocidad de muestreo.

Tomando como ideal al convertidor A/D, el único error presente (ruido) es la cuantización. Esto quiere decir que el modelo de ruido de cuantización es aceptado como un ruido blanco[7] aditivo en varios casos prácticos, siendo más común si hay una resolución alta (>6 bits). Por lo que *ENOB* es expresado en función de *SNR* como:

$$ENOB=\frac{SNR-1.76}{6.02} \tag{9}$$

A pesar de ser una especificación dinámica, en ella quedan reflejadas todas las imperfecciones del sistema, es decir, engloba todas las fuentes de ruido, es por esto que se puede decir que ésta es la mejor manera de comprobar el rendimiento del convertidor [18].

2.3.6 Consumo de Potencia

El consumo de Potencia en un circuito proviene de dos fuentes, el consumo estático y el consumo dinámico. El consumo estático es el consumo que se produce debido a corrientes de fuga existentes en los transistores y está dado por el producto del voltaje de alimentación V_{dd} y la corriente que circula en el circuito $I_{estático}$ [20].

$$P_{estático} = V_{dd} * I_{estático} \tag{10}$$

Idealmente el consumo debería de ser cero debido a que en estado estático el circuito se encuentra apagado y por lo tanto no existe una relación entre V_{dd} y *gnd*. Éste consumo es generalmente despreciable si no se trabaja a temperaturas excesivamente altas, ya que dichas pérdidas dependen de la temperatura de forma exponencial. Por otra parte, el valor de este consumo depende de características de la tecnología que se emplea, el número de transistores y la temperatura de funcionamiento del circuito.

[7] El ruido blanco es una señal no correlativa, es decir, en el eje del tiempo la señal toma valores sin ninguna relación unos con otros.

El **consumo dinámico** se produce debido a la carga y descarga de la capacidad de los transistores y las conexiones, y depende de la actividad del circuito. Es decir ocurre únicamente cuando las puertas están conmutando, por este motivo el consumo de potencia es proporcional a la frecuencia de conmutación y cuanto mayor sea el número de conmutaciones mayor será el consumo de potencia dinámico, esto se representa matemáticamente como:

$$P_{\text{dinámica}}=\alpha*V^2_{dd}*f*C \tag{11}$$

Donde α es un factor de actividad entre 0 y 1, V_{dd} es el voltaje de alimentación, f la frecuencia de operación y C la capacitancia de la carga. La potencia dinámica tiene dos componentes, la potencia de conmutación y la de carga. La primera es debida a las corrientes que van desde las fuentes de alimentación a tierra cuando el transistor cambia de estado, mientras que la de carga se debe a la corriente necesaria para cargar las capacidades de los elementos conectados a la salida.

$$P_{\text{dinámica}}=P_{\text{conmutación}}+P_{\text{carga}} \tag{12}$$

Por lo ya mencionado se debe tomar en cuenta el consumo de potencia de los convertidores para ser alimentados ya sea por baterías o directamente de la toma de corriente, para así obtener un mayor rendimiento del circuito.

2.4 Aplicación de los Moduladores

Los Moduladores A/D ΣΔ se han destaco por tener alta resolución (mayor a 16 bits) y alta estabilidad con frecuencias de muestreo de 4ksps (muestras por segundo) a 10Msps siendo su consumo de potencia baja y por consecuencia es un modulador de costo moderado [14]. Estos moduladores son usados más comúnmente en audio, video, imagen, señales inalámbricas, etc. De modo que son utilizadas en unidades de instrumentación como los cromatógrafos de gas que se utilizan en las industrias del petróleo y gas que normalmente exigen una precisión de tantos bits como sea posible, de igual manera son utilizadas en instrumentación médica, por mencionar algunos ejemplos.

Los buenos resultados de estos convertidores los han hecho cada vez más populares, por lo que están siendo utilizados en más áreas del conocimiento (telecomunicaciones, medicina, cómputo, etc).

2.5 EL FPAA

Las Matrices analógicas programables en campo (FPAAs, en inglés) incluyen un conjunto de bloques analógicos configurables (CABs, en inglés) y una red de enrutamiento usada para pasar señales entre los bloques de

construcción y una colección de elementos de memoria usados para definir tanto la función como la estructura. Dependiendo de la topología los CABs se usan para implementar una serie de funciones de procesamiento de señales analógicas como suma, resta, multiplicación, división comparación, logarítmicas entre otras. Un diagrama conceptual de un FPAA con CABs y red de enrutamiento se puede ver en la Figura 2.13 [5], [21].

Los FPAA actualmente disponibles varían en arquitectura y diseño de interconexión, a menudo son limitados en tamaño y flexibilidad, tal es el caso de los FPAA de los proveedores Anadigm [22]. En la Figura 2.14 se muestra la arquitectura interna del FPAA AN231E04d Anadigm. Consta de una matriz 2x2 de bloques analógicos (CABs) totalmente configurables, rodeados por recursos de interconexión programables y celdas de entrada /salida analógica con elementos activos [23].

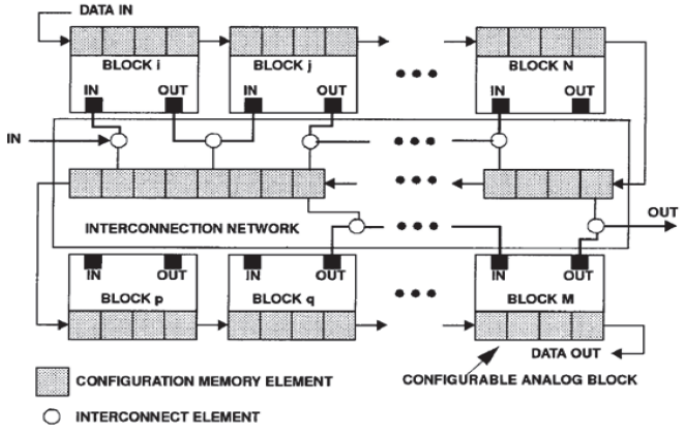


Figura 2.13 Diagrama conceptual de FPAA [5].

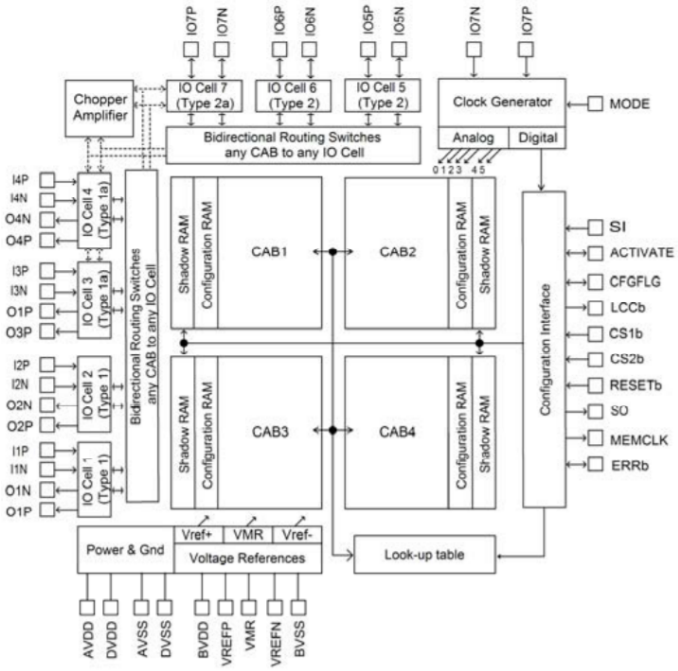


Figura 2.14 Arquitectura interna del FPAA AN231E04d Anadigm más popular con los cuatro bloques analógicos configurables [23].

Los circuitos analógicos y digitales han ayudado a desarrollar diferentes aplicaciones funcionales en la industria de la electrónica, así como en la investigación. En los sistemas reconfigurables, la eficacia y la calidad del sistema de circuitos analógicos pueden mantenerse y modificarse mediante circuitos o cambiar los valores de los elementos individuales. Sin embargo, hay ventajas adicionales de las aplicaciones de circuitos analógicos programables, ya que se pueden producir sistemas más compactos, más fiables y más flexibles con un mejor rendimiento. Es especialmente beneficioso si para la programación modificamos la función de un circuito analógico programable ya sea dando una nueva topología o un nuevo parámetro de componente utilizando la flexibilidad del microcontrolador. Los FPAAs pueden diseñarse usando dos tecnologías: en Tiempo Discreto (Condensador conmutado) y Tiempo Continuo.

2.5.1 FPAA Tiempo Discreto (Condensador conmutado)

Los circuitos de condensador conmutado son muy utilizados para el desarrollo de bloques de procesamiento de señales analógicas en circuitos integrados MOS, como: funciones de filtraje, oscilador de voltaje controlado, moduladores, etapas de ganancia, etc. El conmutador y capacitor forman un tipo de registro análogo y la ruta de señal del sistema está dividida por estos registros. Los elementos computacionales básicos son usualmente amplificadores operacionales y registros análogos, que sintetizan una resistencia lineal cuyo valor es determinado por la razón de conmutación y el valor del capacitor. La respuesta de frecuencia se determina en función de la

frecuencia del reloj, donde el cristal oscilador es usado para ajustar la frecuencia del reloj [24].

Interruptores en circuitos de capacitores conmutados son usados como elementos de muestreo que pueden ser activados por fases de reloj sin-superposición. En la Figura 2.15 podemos observar condensadores e interruptores que se utilizan en el diseño de los FPAA de tiempo discreto, donde la capacidad de programación dentro de los diseños de capacitores conmutados se logra usualmente usando una serie de condensadores como se muestra. La capacitancia efectiva en cada interruptor puede variar ajustando las n celdas de memorias digitales que son controlados por los conmutadores S_1 – S_n.

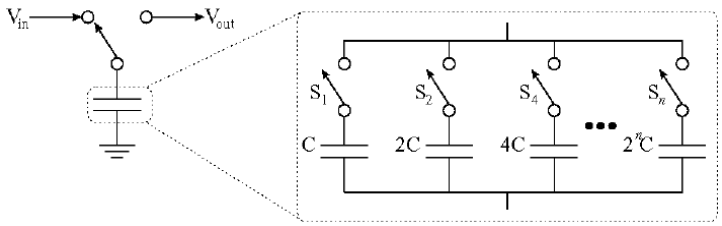


Figura 2.15 Condensadores conmutados para el diseño de un FPAA de Tiempo discreto.

2.5.2 FPAA Tiempo Continuo

En esta tecnología los interruptores son controlados generalmente por los registros digitales, donde pueden ser cargados por un regulador externo, Op-Amps combinados con redes pasivas para servir como bloque funcional, esto permite que el FPAA se pueda configurar para diversos diseños. Volviéndolo así ventajoso ya que no necesitan filtros antialiasing. En comparación con la tecnología de Tiempo discreto en esta pueden usarse procesos de diseño comunes y anchos de banda de señales grandes que pueden ser soportados con un rendimiento predecible. La desventaja es que en cada celda de conexión hay un Rango de ajuste limitado y se requiere de memorias mul-tivaloradas [5]- [24].

2.6 SIMULINK

La velocidad en el crecimiento y desarrollo en la tecnología electrónica ha conllevado nuevos retos y tareas complejas. A este respecto existen herramientas de computo tales como *Agilent Advanced Design System*, CADENCE, SIMULINK, entre otras, que han ayudado a enfrentar dichos retos. SIMULINK "*es una herramienta para simular sistemas dinámicos con una interfaz gráfica especialmente desarrollada para este propósito*" [10]. La Figura 2.16 ilustra un ejemplo sencillo del ambiente de simulación de SIMULINK. Cabe mencionar que SIMULINK es una herramienta de MATLAB. Como se mencionó

anteriormente, se pueden simular modelos o sistemas con cierto grado de abstracción. SIMULINK cuenta con soporte técnico, además de proporcionar herramientas para la ejecución remota de programas, donde se pueden controlar en “tiempo real” algunas tareas. SIMULINK es un entorno de programación de nivel de abstracción más alto que el lenguaje interpretado por Matlab. SIMULINK genera archivos con extensión *mdl* (de "model"), que son guardados en el espacio de almacenamiento de resultados de simulación de MATLAB. La simulación es realizada por el programa calculando el valor de las señales involucradas a intervalos muy pequeños de tiempo. Las operaciones que se pueden realizar son aritméticas, así como la solución de ecuaciones diferenciales [25].

Debido a las características descritas anteriormente, SIMULINK es una herramienta muy popular tanto en la industria como en la academia, ya que permite desarrollar modelos comportamentales de sistemas dinámicos y trasladarlos a algún sistema embebido tal como un microcontrolador o un FPGA [26]. Por lo tanto, es posible realizar prototipados rápidos y de bajo costo comparados con un circuito integrado.

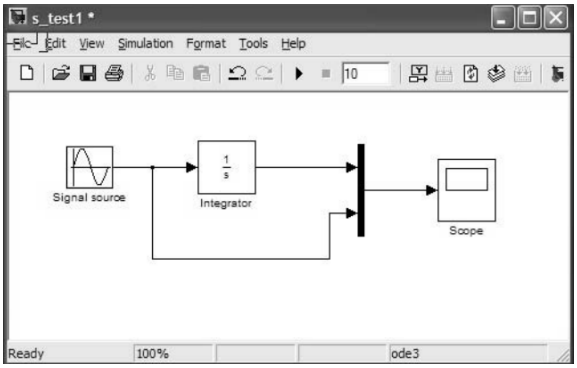


Figura 2.16 Representación de un diagrama de bloques en SIMULINK.

2.7 SIMSIDES

SIMSIDES (SIMULINK-based Sigma-Delta Simulator, en inglés), es un simulador comportamental en el dominio del tiempo para Moduladores ΣΔ, desarrollado como una herramienta en el entorno de SIMULINK/MATLAB. Puede ser utilizado para simular las diferentes arquitecturas de moduladores-ΣΔ tanto en tiempo continuo como en tiempo discreto. Contiene una lista completa de bloques construidos de moduladores- ΣΔ (Integradores, resonadores, cuantizadores, etc.), incluidas en la herramienta. Los modelos comportamentales de los bloques toman en cuenta los más críticos mecanismos de error como el de los capacitores conmutados. Los modelos son confiables ya que fueron validados por medio de simulaciones eléctricas a nivel de transistor y por mediciones experimentales tomadas de una serie de prototipos en circuito integrado [27].

En la Figura 2.17 se muestra la interfaz de usuario de la herramienta, esta tiene una gran flexibilidad para la extensión de nuevos modelos y bloques de construcción, y un potente conjunto de rutinas de procesamiento de señales. SIMSIDES cuenta aproximadamente con más de 150 *S-functions* y 520 modelos comportamentales.

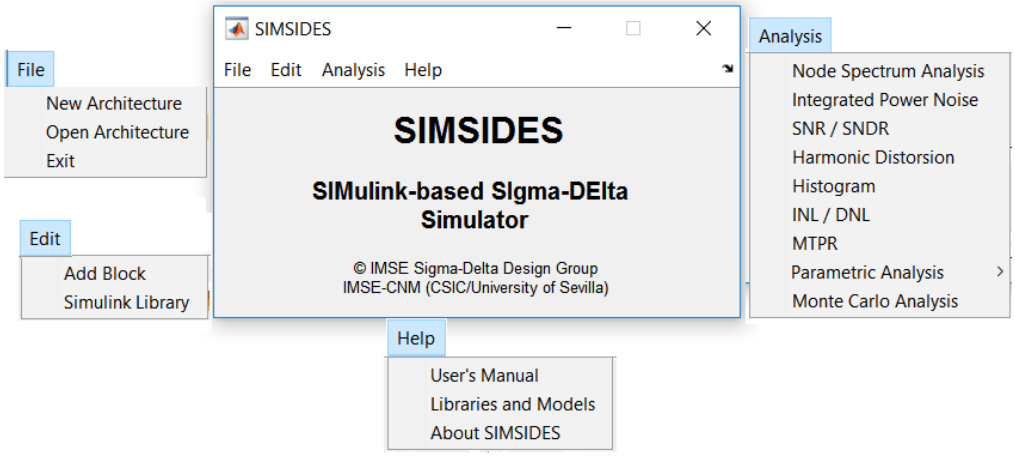


Figura 2.17 Interfaz de la herramienta SIMESIDES.

SIMSIDES cuenta con librerías ideales y reales, las primeras contienen *S-functions* con modelos ideales esto quiere decir que son modelos sin contemplar errores del mecanismo, como ruido térmico generado por los conmutadores, error de ganancia, capacitancias parásitas, entre otras. Por otra parte, la librería real contiene *S-functions* de modelos comportamentales que incluyen los mecanismos de error más críticos (Tabla 2.2) que alteran el rendimiento de los moduladores- ΣΔ.

Tabla 2.2 Circuitos y mecanismos de error modelados en SIMSIDES [16]

Circuit Technique	Building Block	Error Mechanism
Switched-Capacitor	Amplifiers	Finite and Nonlinear DC gain, incomplete settling error, output swing limitation, thermal noise
	Switches	Thermal noise, finite and nonlinear switch on-resistance
	Capacitors	Mismatch, nonlinearities, parasitic capacitances
Switched-Current	Memory cells and Integrators	Linear and nonlinear gain error, thermal noise, finite output−input conductance ratio error, charge injection error, incomplete settling error
Continuous-Time	Integrators	Finite and nonlinear DC gain, nonlinear transconductance, thermal noise, output swing limitation, transient response
All circuit techniques	Clock generator	Clock jitter
	Comparators	Hysteresis and offset
	Quantizers/DACs	Nonlinearity (INL), gain error, excess loop delay, offset

Las librerías que contienen integradores y resonadores se subdividen en varias sub-bibliotecas específicas, contienen a su vez modelos de bloques de construcción correspondientes a diferentes implementaciones de niveles de circuito. Por ejemplo, los integradores con condensadores conmutados (SC) se subdividen en integradores FE (Forward-Euler) y LD (Lossless discrete); integradores de CT son subdivididos en Gm-C, active-RC, etc.

2.8 Trabajos en diseños de software

En el año 2011, Raphael da Costa Neves y Alessandro Girardi [4] desarrollaron una metodología para un prototipado rápido de moduladores ΣΔ usando arreglos analógicos reconfigurable. Se aborda el diseño de un modulador de 2° orden ΣΔ en una tarjeta FPAA. El diseño se realizó en SIMULINK/MATLAB como se puede observar en la Figura 13, bajando el diseño a una tarjeta FPAA. Los resultados obtenidos de dicha tarjeta fueron comparados con diseños personalizados, obteniendo resultados similares, por lo que se concluye, que el proyecto ahorra tiempo y costo en el diseño de un circuito en específico.

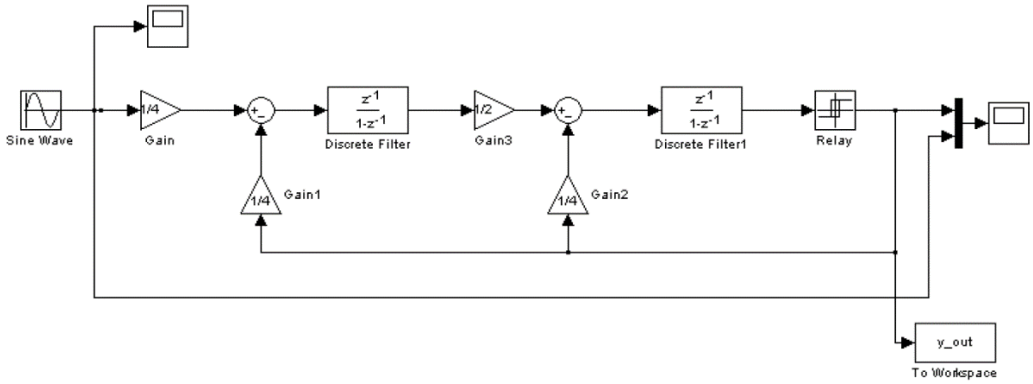


Figura 2.13 Diagrama del modulador de segundo orden ΣΔ implementado en Simulink-Matlab [4]

En el año 2012, José M. de la Rosa [28] propone un modelo de comportamiento y simulación para aprender circuitos de comunicación y sistemas. Toman dos casos de estudios a considerar, en el primero se da un enfoque de cómo enseñar a estudiantes el diseño de convertidores digital-analógico, para los principales medios de comportamiento de la simulación. El segundo muestra el aprendizaje de la planeación del sistema de procesos de radio frecuencia en transceptores inalámbricos. Ambos casos son modelos compartamentales implementados en Matlab/Simulink. Los resultados son sustentados con más de diez años de experiencia y se ha demostrado ser un método aceptable por mayoría de los estudiantes como se muestra en la gráfica de la Figura 2.14, que incluye el aprendizaje basado en simulación.

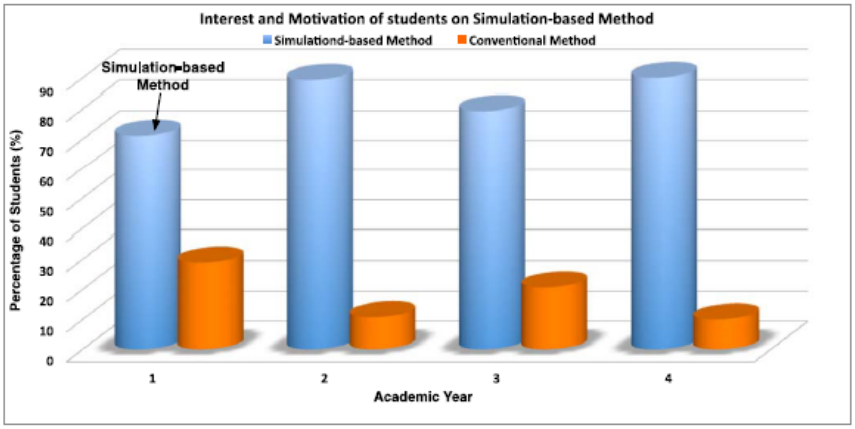


Figura 2.14 Retroalimentación y los resultados de los estudiantes: Interés y motivación para los contenidos basados en la simulación. [28]

En el año 2014, P.J. Obeso-Rodelo et al [29] propusieron una metodología de diseño e implementación de un oscilador caótico de Chua en un FPGA. Implementaron un oscilador caótico específicamente un circuito tipo Chua usando la herramienta Simulink/Matlab en una tarjeta FPGA. Realizando la simulación en SIMULINK/MATLAB, donde comparando los resultados de la simulación en SIMULIK como se muestra en la Figura 2.15 con los resultados obtenidos de la tarjeta FPGA como se muestra en la Figura 2.16, son representaciones gráficas del sistema que muestran cómo los estados de cada variable describen un patrón complejo y no repetitivo lo cual representa la aleatoriedad del sistema que era el resultado esperado.

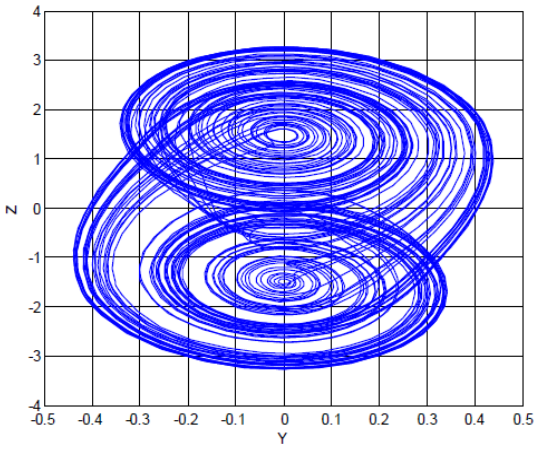


Figura 2.15 Atractor y-z en 2D en Matlab/Simulink [29].

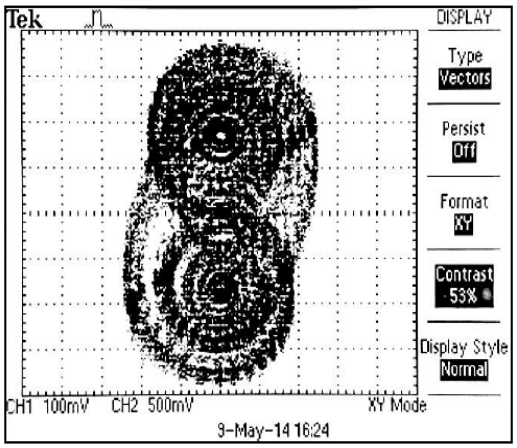


Figura 2.16 Atractor y-z en 2D en Osciloscopio [29].

En el año 2016 Krzysztof Wawryn, Robert Suszynski y Jacek Marciniak [30] proponen un enfoque de hardware basado en la toma de redes neuronales artificiales para clasificar un valor calorífico de un combustible de carbón en cámaras de combustión. Es decir, se analizan los parámetros del proceso de combustión medida en la cámara. Los parámetros medidos se han utilizado para entrenar pesos de la red neuronal con una ayuda del programa de MATLAB. El ganador toma toda la fórmula que se ha utilizado para entrenar pesos sinápticos. El método se basa en que el ganador toma todas las redes neuronales artificiales (RNA) implementadas en un dispositivo de campo analógico (FPAA). Los métodos de ANN se utilizan con éxito para resolver problemas de clasificación multidimensional. El dispositivo FPAA proporciona varias ventajas para construir sistemas de hardware, es decir, programación, procesamiento paralelo y creación de prototipos.

En el año 2017, Jennifer Hasler, Aishwarya Natarajan, Sahil Shah, y Sihwan Kim [31] implementaron una clase de nivel junior que va desde el concepto clásico de circuito discreto hasta el diseño de nivel de sistema. Se utilizó Matrices analógicas programables de campo a gran escala (FPAA) (ECE 3400) para los proyectos prácticos. La estructura FPAA sólo requiere una placa conectada a través de USB a una computadora portátil del estudiante con la computadora virtual de código abierto. Este trabajo tiene como resultado los datos de implementación, análisis y evaluación temprana para la clase de primer semestre. Donde los estudiantes pueden diseñar un componente del sistema (por ejemplo, un ADC de Rampa).

Capítulo 3 Metodología

Para el desarrollo de la herramienta de cómputo para el prototipado de los moduladores de datos analógico-digital ΣΔ en SIMULINK/MATLAB se tomó como referencia el método de desarrollo ágil, siguiendo los pasos de análisis de requerimientos (convertidores ΣΔ), diseño (modelo comportamental), programación (modelado a nivel sistema), pruebas (simulación en Matlab) estos pasos se seguirán de acuerdo a la Figura3.1.

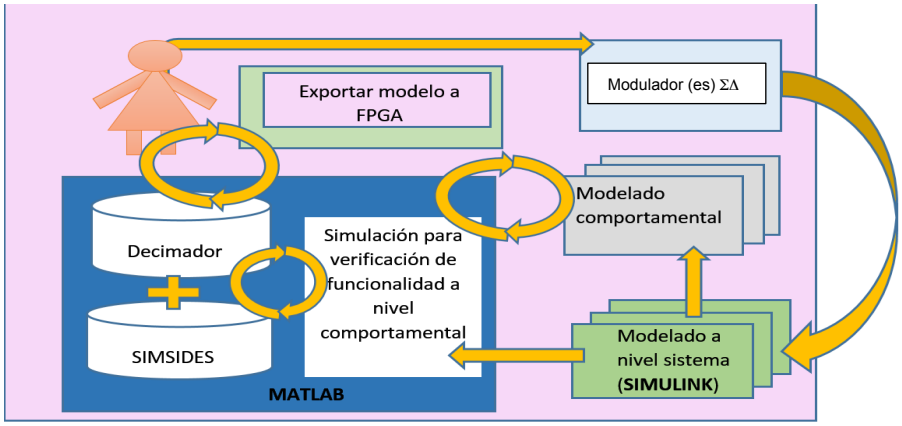


Figura 3.1 Metodología propuesta para el desarrollo del proyecto.

3.1 Análisis de requerimientos.

Se requiere hacer una herramienta en Simulink/Matlab para el prototipado de los moduladores de datos analógico-digital ΣΔ y bajarlo a un embebido para verificar su correcto funcionamiento.

Para esta etapa se recolectó de acuerdo a la literatura los diseños de los convertidores A/D ΣΔ y las métricas (*DR*, *SNR*,*SFDR*,Potencia, frecuencia de

muestreo, etc.) correspondientes a los diferentes bloques de las arquitecturas de cada uno de ellos como se muestran en las Tablas 3.1.

La herramienta que se ocupó para el desarrollo de este proyecto es Simulink/Matlab 2016a, SIMSIDES, la tarjeta FPAA Anadigm AN231E04, además se utilizó el kit de desarrollo AN231K04-DVLP3 que es una plataforma muy amigable de utilizar para la implementación y pruebas de circuitos analógicos. Contiene una interfaz PC serial para poder descargar los circuitos diseñados en el software AnadimgDesigner2 (AD2), de la misma forma, se utilizó la plataforma NI Elvis II para generar la señal sinusoidal.

Tablas 3.1 Valores de métricas de moduladores ideales y reales a) [32], b) [33] y (c) [32].

ΣΔ Modulator Parameter	*SNDR [dB]*	*Resolution [bits]*
Ideal modulator	101.5	16.56
Sampling jitter ($\Delta\tau$ = 8 ns)	96.7	15.76
Switches (kT/C) noise (Cs = 5 pF)	98.7	16.11
Input-referred op-amp noise (Vn = 50 μVrms)	95.5	15.58
Finite dc gain ($H_0 = 1{\cdot}10^3$)	101	16.48
Finite bandwidth (GBW = 100 MHz)	86.5	14.07
Slew-rate (SR = 18 V/μs)	76.1	12.36
Saturation voltages ($Vmax$ = ± 1.34 V)	96.8	15.79

a)

Block	Parameter	Initial value	Weight
1.Integrator	OpAmp gain	82 dB	1.9642
	OpAmp slew rate	18 V/μs	0.1043
	OpAmp bandwidth	22 MHz	0.8526
	OpAmp offset	8 μV	1.2732
2.Integrator	OpAmp gain	82 dB	0.7067
	OpAmp slew rate	18 V/μs	n.s.
	OpAmp bandwidth	22 MHz	n.s.
	OpAmp offset	8 μV	1.3514
ADC	Level error	1 mV	0.1941
	Hysteresis	1 mV	0.1040
DAC	Output level error (-5%/+10%)	2.5 V	5.0182

b)

Parameter	*Value*
Signal bandwidth	BW = 22.05 KHz
Sampling frequency	F_S = 11.2896 MHz
Oversampling ratio	R = 256
Samples number	N = 65536
Integrator gain	$b = b_2 = 0.5$

c)

3.2 Modelado comportamental

Para el desarrollo de esta etapa, se utilizó la herramienta SIMSIDES de donde se tomaron bloques (filtros, cuantizadores, etc.) que conforman las diferentes topologías de los moduladores A/D $\Sigma\Delta$ y se programaron los parámetros.

3.2.1 Integradores

En la Figura 3.2 se muestra un esquema conceptual (unipolar) de un integrador SC FE o de tiempo discreto (TD) (a) y un integrador de tipo Gm-C o de tiempo continuo (TC) (b). Considerando al circuito como ideal, con un voltaje de salida v_o de la Figura 3.1a y está dada por la siguiente ecuación diferencial finita:

$$v_o(nT_s) = (C_s / C_i)\, v_i[(n-1)T_s] + v_o[(n-1)T_s] \tag{3.1}$$

Donde v_i es la entrada del voltaje, C_s y C_i son los capacitores de muestreo y de integración respectivamente y nT_s representa el tiempo de muestreo instantáneo enésima y T_s el periodo de muestreo.

La funcionalidad ideal de la Figura 3.2b puede ser expresada matemáticamente por la siguiente ecuación diferencial:

$$g_m v_i(t) = C \frac{dv_o(t)}{dt} \tag{3.2}$$

Donde g_m es la transconductancia y C es la integración de la capacitancia.

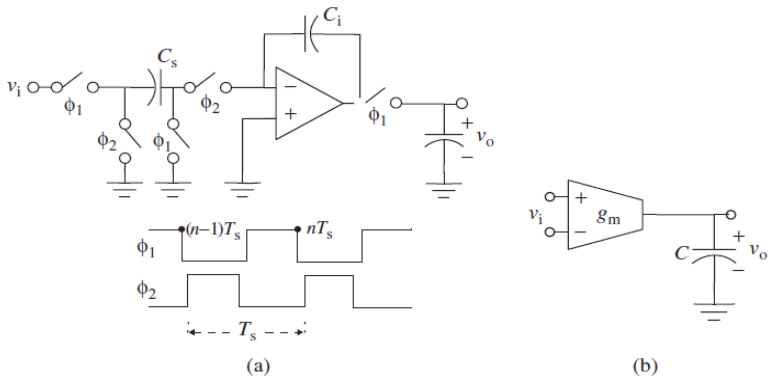


Figura 3.2 Esquema conceptual de: a) integrador de TD y b) integrador de TC.

Los modelos descritos para los integradores en el dominio del tiempo pueden ser implementados en el dominio de la frecuencia usando librerías de bloques en Simulink como se muestra en la Figura 3.3. Sin embargo esto no es eficiente en situaciones prácticas ya que el rendimiento de los moduladores disminuye por el efecto de los errores del nivel del circuito. La mayoría de los

errores de circuito se modelan de una manera más precisa si los modelos comportamentales son descritos en el dominio del tiempo en lugar del dominio de la frecuencia [16].

Consideremos el efecto de la ganancia finita de DC de un integrador OTA (Operational Transconductance Amplifier) como se muestra en la Figura 3.4, donde A_v representa la ganancia de tensión finita DC del amplificador operacional en el integrador SC FE y g_o es la conductancia de salida finita del transconductor en el integrador Gm-C, de manera que la ganancia finita de DC está dada por g_m/g_o. Matemáticamente se describe el comportamiento de los integradores en el dominio del tiempo como:

$$v_o(nTs) = \frac{1+\mu}{1+(1+g)\mu} v_o[(n-1)Ts] + \frac{g}{1+(1+g)\mu} v_i[(n-1)Ts] \quad (3.3)$$

$$g_m v_i(t) = C\frac{dv_o(t)}{dt} + g_o v_o(t)$$

Donde $\mu \equiv 1/A_v$ y $g = C_s/C_i$

Tomando en cuenta los efectos no lineales, las ecuaciones de balance de carga se transforman en los siguientes:

$$v_{C_s,\varphi 1} = -v_i[(n-1)T_s]$$

$$v_{C_i,\varphi 1} = -[1 + 1/A_{v(n-3/2)}] \cdot v_{o1}[(n-1)T_s] = -(1+\mu) \cdot v_{o1}[(n-3/2)T_s]$$

$$v_{C_s,\varphi 2} = -[1/A_{v(n-1/2)}] \cdot v_{o1}[(n-1/2)T_s] \quad (3.4)$$

$$v_{C_i,\varphi 2} = -[1 + A_{v(n-1/2)}] \cdot v_{o1}[(n-1/2)T_s]$$

Donde $A_{v(n-1/2)} = A_v(v_{o1}[(n-1/2)T_s])$ y $A_{v(n-3/2)} = A_v(vo1[(n-3/2)T_s])$.
La ecuación en diferencias finitas no lineal que describe el comportamiento del integrador se puede escribir como:

$$\left(1 + \frac{1+g}{A_{v(n-1/2)}}\right).v_{o1}\left[\left(n-\frac{1}{2}\right)T_s\right] = g.v_i[(n-1)T_s] + \left(1 + \frac{1}{A_{v(n-3/2)}}\right).v_{o1}[(n-3/2)T_s] \quad (3.5)$$

Se debe tomar en cuenta que se necesita un procedimiento iterativo para calcular el modelo de comportamiento en la ecuación 3.5, debido a que la tensión de salida del amplificador operacional V_{O1} (nT_s) depende del amplificador de ganancia no lineal, lo que resulta en cambios con V_{O1} (nT_s).

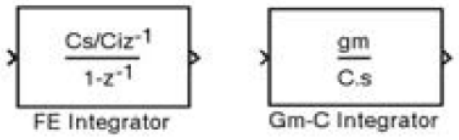


Figura 3.3 Modelos de integradores ideales en Simulink de la Figura 3.1.

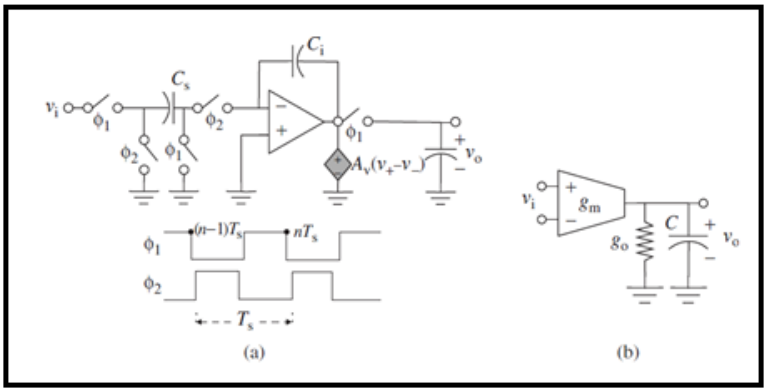


Figura 3.4 Esquema OTA de ganancia finita de CD [16].

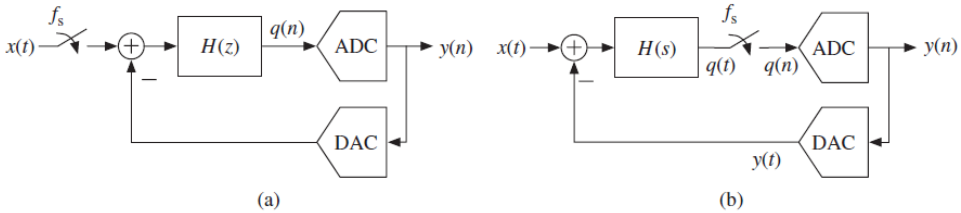


Figura 3.5 Diagramas de bloques de moduladores SD de tiempo continuo (TC) y tiempo discreto (TD) a) modulador TD, b) modulador TC [16].

3.3 Modelos y librerías

En esta etapa se programaron cada uno de los bloques de acuerdo a las métricas de cada modulador ΣΔ.

Con el apoyo de la herramienta SIMSIDES se programaron en SIMULINK/MATLAB diferentes filtros de tiempo continuo (TC) y tiempo discreto (TD) de las diferentes arquitecturas de dichos convertidores, como se muestran en las Figuras 3.5.

3.3.1 Integradores SC Reales

En la practica el rendimiento de los integradores SC (condensadores conmutados) se ve perjudicado por un grupo grande de errores de mecanismos, en la Figura 3.6 se observa el diagrama de flujo de un integrador SC FE (Forward-Euler) donde de forma iterativa se calcula el funcionamiento comportamental correcto, para obtener el comportamiento preciso se toma en

cuenta la contribución de errores del circuito principal, así como las fases del reloj.

El modelo comienza cargando los valores de los parámetros requeridos como la señal de entrada y las condiciones iniciales, es decir, los voltajes en los nodos internos del integrador (incluidos los valores almacenados en el condensador de muestreo y de integración) almacenados en el reloj del anterior período y que generalmente se establece en cero al comienzo de una simulación.

Antes de comenzar a calcular el comportamiento del modelo se realizan algunos cálculos iniciales, como: el ruido equivalente referido a la entrada, el valor real del peso del integrador debido a la falta de coincidencia del condensador y las capacitancias parásitas en diferentes nodos del integrador durante el muestreo y fases de reloj de integración.

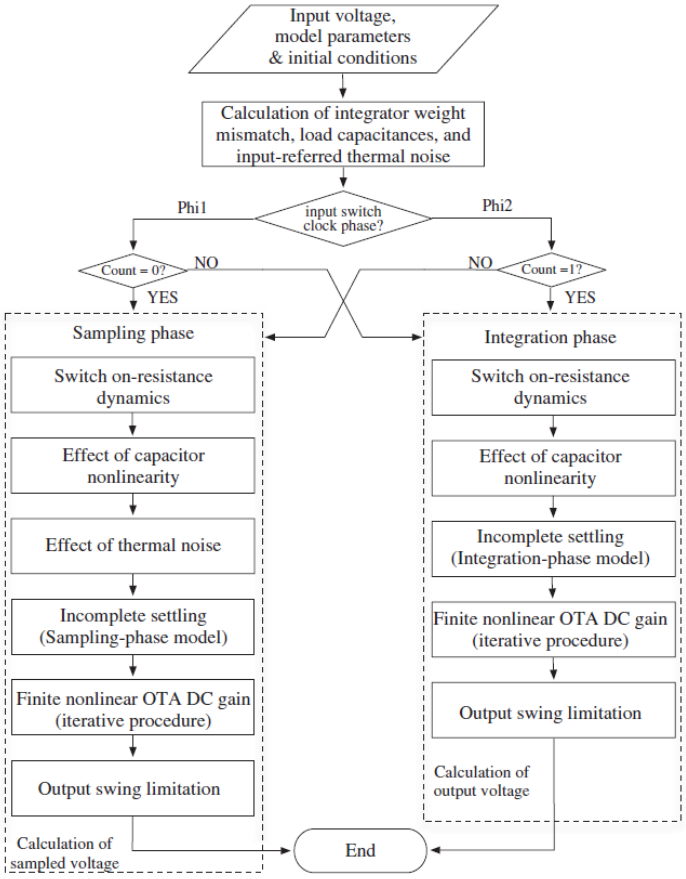


Figura 3.6. Diagrama de flujo del modelo computacional del integrador SC FE [34].

Tabla 3.2. Todos los modelos de integrador SC disponibles en SIMSIDES [27].

Ideal Libraries	**Sublibraries**	**Building Blocks**
Integrators	–	Ideal DT/CT integrators
Resonators	–	Ideal resonators
Quantizers & Comparators	–	Ideal quantizers
D/A Converters	–	Ideal DACs
Real Libraries	**Sublibraries**	**Building Blocks**
Integrators	SC FE integrators	Forward-Euler SC integrators
	SC LD integrators	Lossless-Direct SC integrators
	SI FE integrators	Forward-Euler SI integrators
	SI LD integrators	Lossless-Direct SI integrators
	gm-C integrators	Gm-C integrators
	gm-MC integrators	Miller OTA integrators
	RC integrators	Active-RC integrators
	MOSFET-C integrators	MOSFET-C integrators
Resonators	SC FE resonators	Resonators based on FE SC integrators
	SC LD resonators	Resonators based on LD SC integratorss
	SI FE resonators	Resonators based on FE SI integrators
	SI LD resonators	Resonators based on LD SI integrators
	gm-C resonators	Resonators based on Gm-C integrators
	gm-LC resonators	Resonators based on Gm-LC integrators
Quantizers & Comparators	–	Nonideal single-bit & multibit quantizers
D/A Converters	–	Nonideal single-bit and multibit DACs
Auxiliary Blocks	–	Adders, latches, DEM blocks, etc

En la Tabla 3.2 se enumera todos los modelos de integradores del modelo SC disponibles en SIMSIDES, incluida una breve descripción de las no idealidades incluidas en cada uno de ellos, tomando en cuenta que los integradores solo corresponden a una rama, sin embargo, los mismos están disponibles para integradores con hasta cuatro ramas de entrada [34]. Por ultimo en la Figura 3.7 se observa una cantidad de diferentes bloques de *S-function* del integrador SC con diferentes números de ramas de entrada y efectos no ideales incluidos en sus modelos, se muestran modelos ideales hasta los más precisos que incluyen todas las no idealidades del circuito.

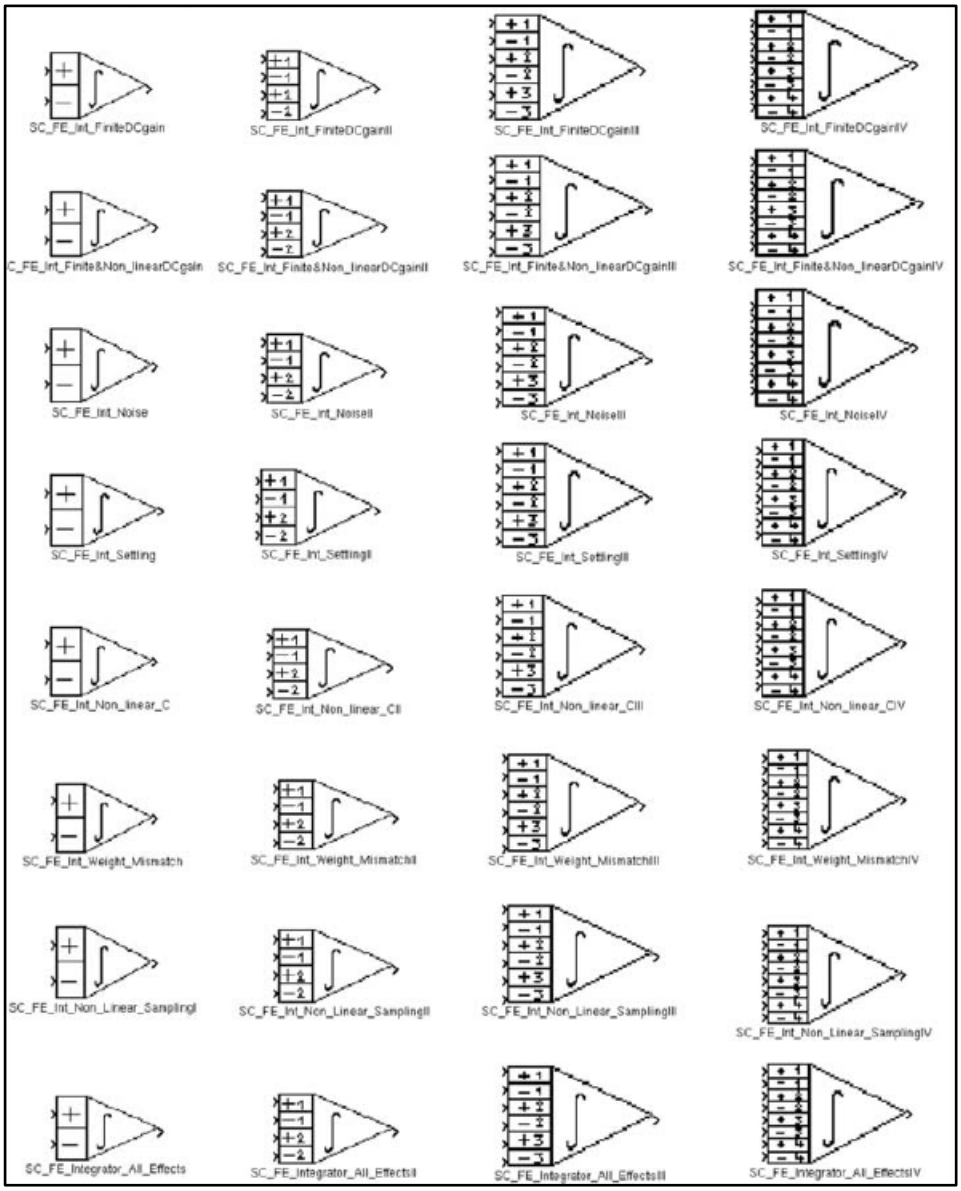


Figura 3.7 Librerías de integradores SC FE de SIMSIDES en Simulin-Matlab [34].

3.3.2 Cuantificadores y comparadores reales

Los bloques de comparadores y cuantizadores reales incluidos en las librerías de SIMSIDES se describen en la Tabla 3.3, sin embargo no solo los parámetros ideales son necesarios para modelar las diferentes irregularidades del circuito si no también son de gran importancia parámetros adicionales, estos parámetros de error se muestran en la Tabla 3.4. Aparte de los comparadores y cuantificadores, hay un bloque de construcción llamado *Real_Sampler*, que se utiliza para modelar los circuitos de S&H (Sample and hold) que están conectados a la entrada de los cuantificadores, integrados en Moduladores- ΣΔ de CT [27].

Tabla 3.3. Modelos de comparadores y cuantizadores incluidos en SIMSIDES [27].

Model name	Circuit effects included
`Real_Comparator_Offset&Hysteresis`	Voltage-mode comparator with offset, (random & deterministic) hysteresis.
`Real_Comparator_Offset&Hysteresis_for_SI`	Current-mode comparator with offset and nonlinearity (INL).
`Real_Multibit_Quantizer`	Voltage-mode multibit quantizer with gain error, offset (random & deterministic) hysteresis.
`Real_Multibit_Quantizer_for_SI`	Current-mode multibit quantizer with gain error, offset, INL, (random & deterministic) hysteresis.
`Real_Multibit_Quantizer_dig_level_SD2`	Voltage-mode multilevel quantizer with gain error, offset, INL, (random & deterministic) hysteresis.
`Real_Sampler`	Sampling & Hold circuit with clock jitter error.

Tabla 3.4. Parámetros de error del modelo real de cuantizadores en SIMSIDES [27].

Parameter name (in alphabetical order)	Brief description
`Gain Error in LSB`	Gain error measured in LSB.
`Jitter typical deviation`	Standard deviation of clock jitter error.
`Kind of Hysteresis`	Comparator hysteresis. It may be either deterministic or random hysteresis.
`INL in LSB`	Integral Nonlinearity error measured in LSB.
`Number of levels`	Number of quantizer levels.
`Offset`	Offset error.
`Offset Error in LSB`	Offset error measured in LSB.
`Seed for random jitter generation`	Seed number used for generating random jitter error.

3.4 DISEÑO (Programación)

Para poder desarrollar este proyecto se tomó como referencia el Modulador $\Sigma\Delta$ de tercer orden desarrollado en SIMSIDES, se utilizó de igual manera la herramienta Matlab R2016a. Para poder hacer uso de la herramienta SIMSIDES es necesario instalar primeramente una *toolbox* que previamente se descargó de la página de SIMSIDES como se mencionó anteriormente. Para realizar la instalación seguimos los siguientes pasos:

1. Establezca la ruta de MATLAB, esto se encuentra en *Archivos de programas* para agregar la carpeta de SIMSIDES.

2. Abrir la herramienta de Matlab, en la pestaña *HOME* seleccionar la opción *Set Path*, posteriormente la opción *Add with Subfolders* e ir a la ruta de Matlab donde se colocó la carpeta de SIMSIDES y seleccionarla. Para guardar los cambios realizados seleccionamos la opción guardar antes de cerrar. En la Figura 3.8 se muestra como quedan cargados los archivos de SIMESIDES en Matlab.

Los pasos uno y dos solo se realizarán la primera vez que se instale SIMSIDES en el disco duro.

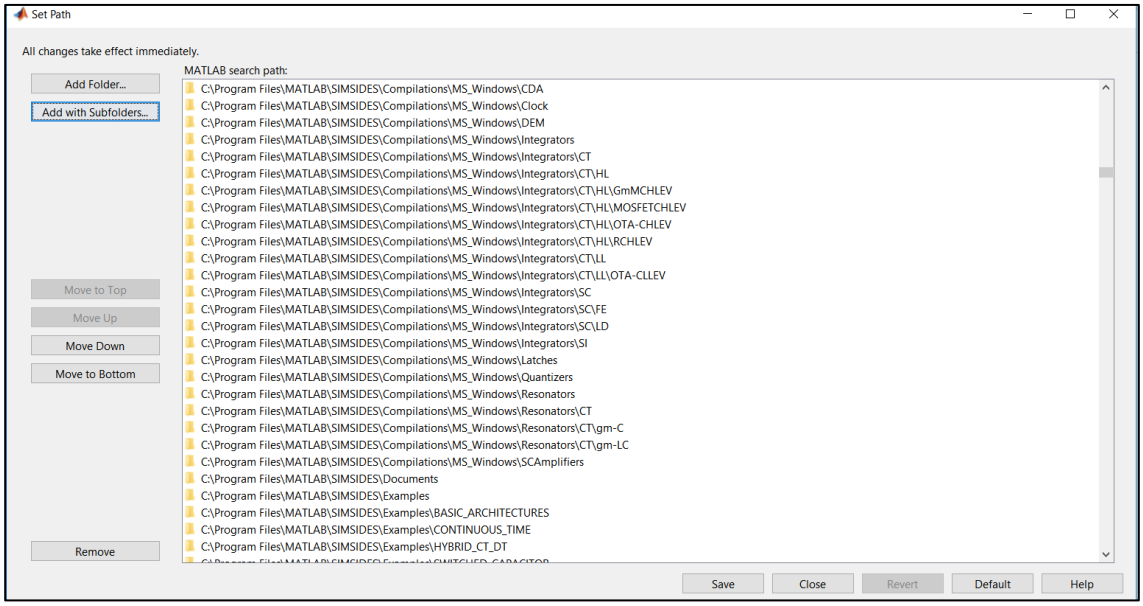


Figura 3.8. Instalación de SIMSIDES

3.4.1 Construcción y edición de arquitecturas de Moduladoles-ΣΔ en SIMSIDES

El Modulador que se realizó como ejemplo para este proyecto se tomó de la guía de SIMSIDES [27] en el que se analizó para mostrar las características principales del Modulador-ΣΔ 2-1 en cascada de tercer orden de tiempo discreto (DT) con cuantificación de un solo bit en ambas etapas.

Para iniciar SIMSIDES, se escribe simsides en la ventana principal de comandos de MATLAB, en la Figura 3.9 se visualiza la ventana principal de la herramienta. Para formar una arquitectura de un Modulador ΣΔ en SIMSIDES, seleccione *File* y luego *New architecture* en el menú principal, y se mostrará una nueva ventana de modelo SIMULINK. Si se requiere una arquitectura de algún Modulador-ΣΔ existente puede ser abierta seleccionando *File -> Open Architecture* como se ilustra en la Figura 3.10.

Figura 3.9 Inicio de SIMSIDES

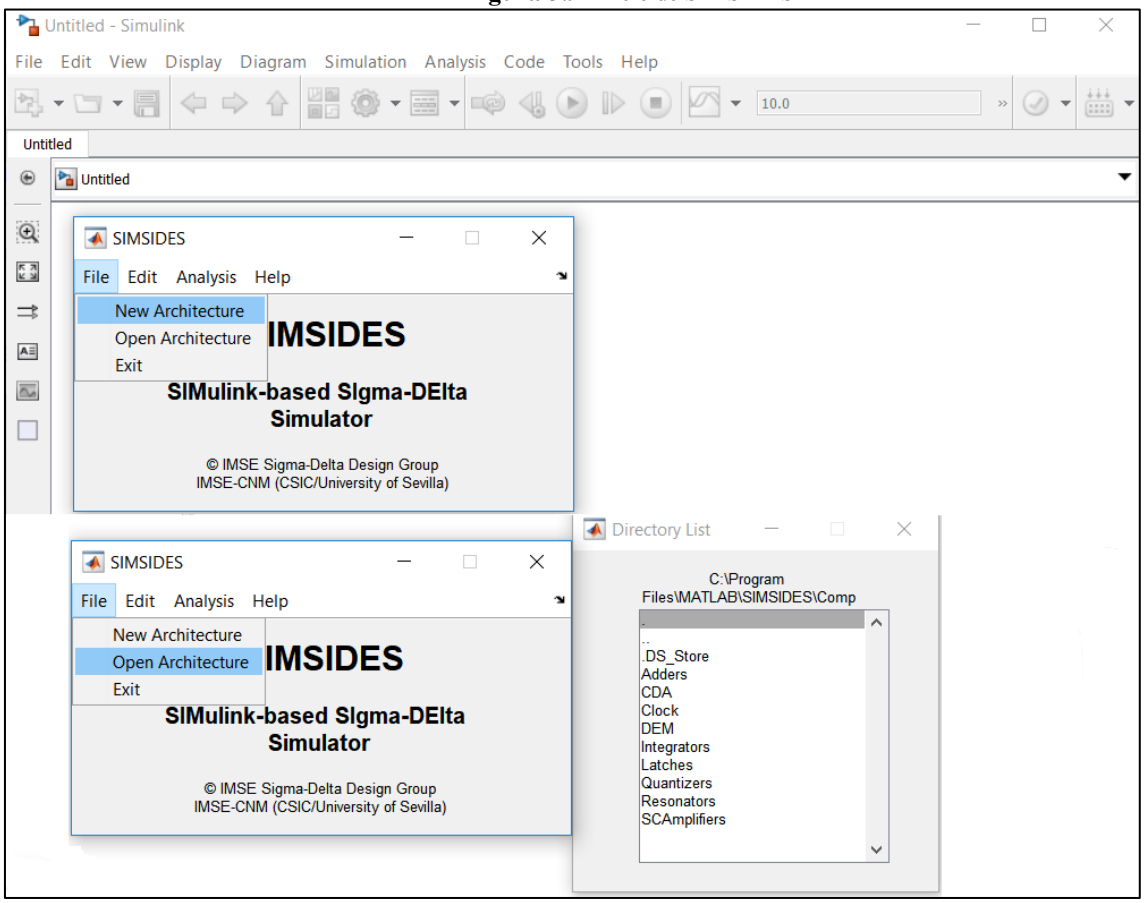


Figura 3.10. Construcción y edición de arquitecturas de Moduladoles-ΣΔ en SIMSIDES

En la Figura 3.11 se muestra el diagrama de bloques del modulador que se implementó usando las librerías de modelos disponibles en SIMSIDES. Para esto se siguieron los siguientes pasos:

1. Seleccione del menú principal de SIMSIDES, *File -> New* architecture e introduzca un nombre para la nueva arquitectura del Modulador.

2. Se incluyen los integradores y los comparadores de las bibliotecas de modelos SIMSIDES. Para hacer esto, seleccione *Edit -> Add*

Block después le desplegará la interfaz donde se escribe la librería a usar ya sea de forma Real o Ideal como se muestra en la Figura 3.12, posteriormente se despliega una nueva ventana con las sub-librerías.

3. Incorporar a la nueva arquitectura simplemente arrastrando y soltando los modelos desde sus correspondientes bibliotecas SIMSIDES. En este ejemplo, los integradores FE de la Figura 3.11 se implementan utilizando los bloques *SC_FE_Integrator_All_Effects* de la libreria Real Integrators, mientras que los cuantificadores de un solo bit se modelan mediante el bloque comparador *Real_Comparator_Offset & Hysteresis* disponible en la libreria *Quantizers & Comparators*.

4. Incorpore los bloques de construcción restantes de la librería de los modelos de SIMULINK. Para hacer esto, vaya a *View -> Library Browser*, Simulink y arrastre los modelos requeridos. En este ejemplo, se requieren los siguientes bloques: bloques de onda sinusoidal y de tierra de la biblioteca de fuentes, bloqueador de retardo de unidad y filtro discreto de la libreria de *Discrete* y la libreria de espacio de trabajo de *Sinks,* Figura 3.12 a.

5. Finalmente, una vez que todos los bloques necesarios se han incluido en la nueva arquitectura, están conectados adecuadamente para implementar la arquitectura requerida del Modulador que se muestra en la Figura 3.11, el M-ΣΔ terminado se muestra en la Figura 3.12 b.

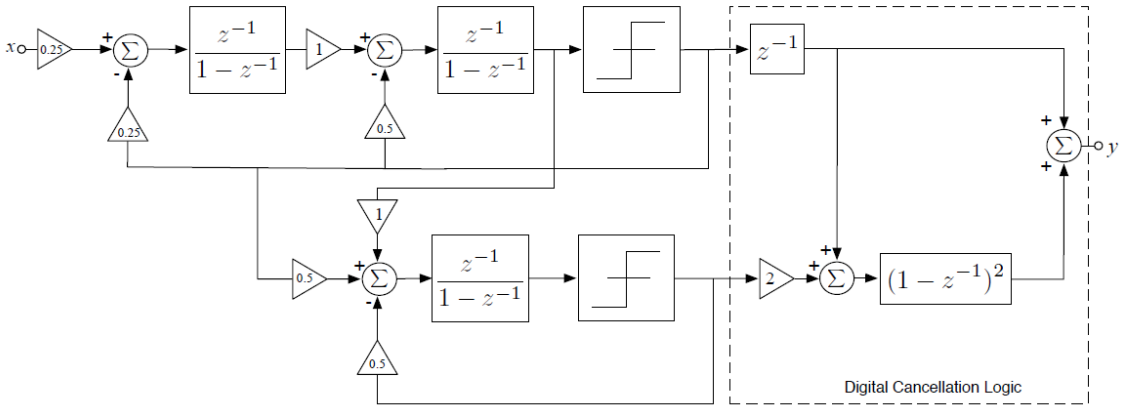


Figura 3.11. Diagrama del M-ΣΔ en cascada 2-1 de DT en bloques de dominio-Z.

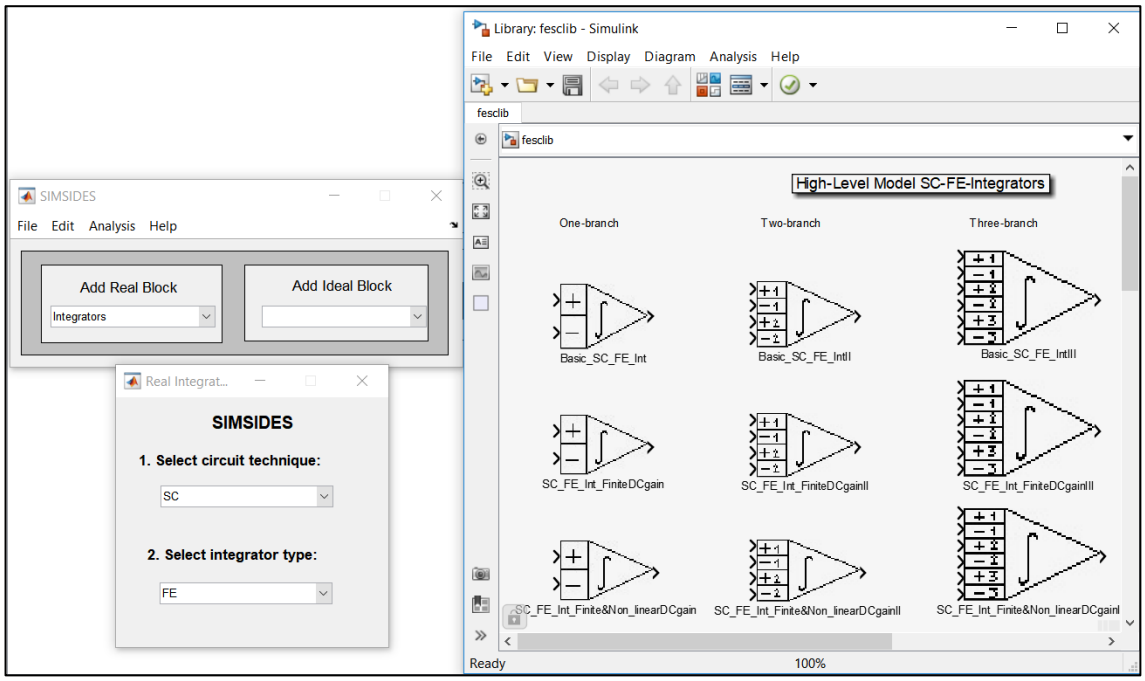
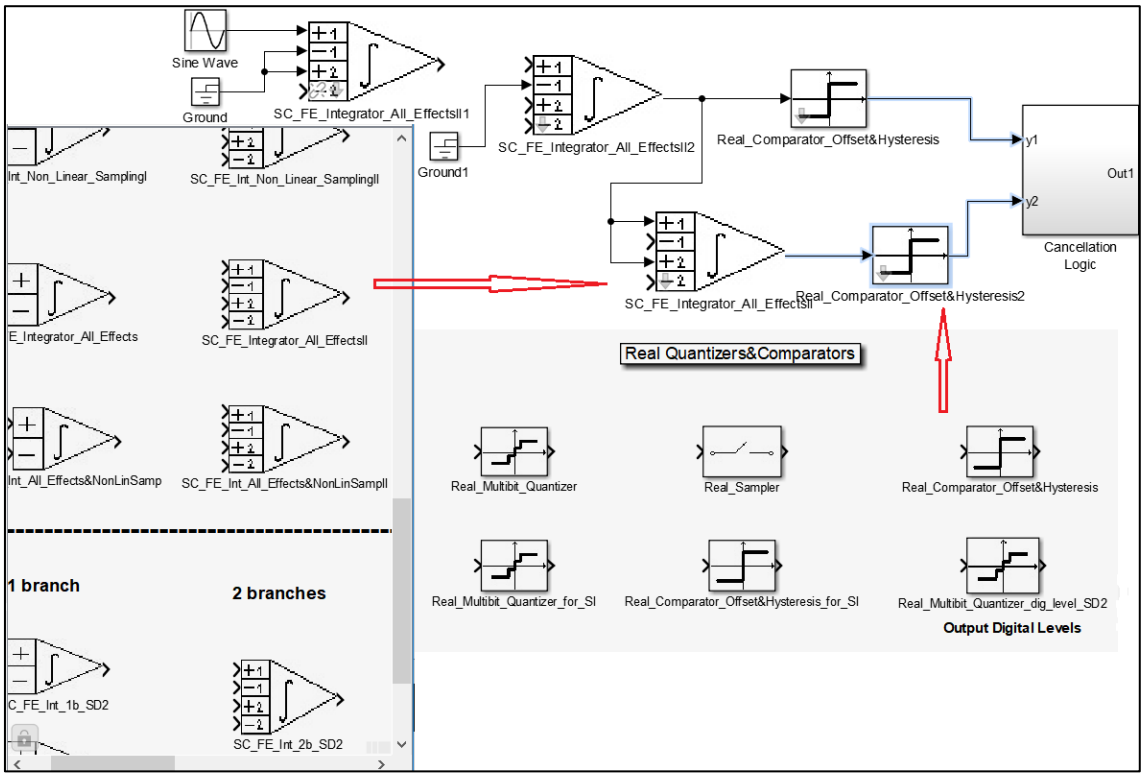


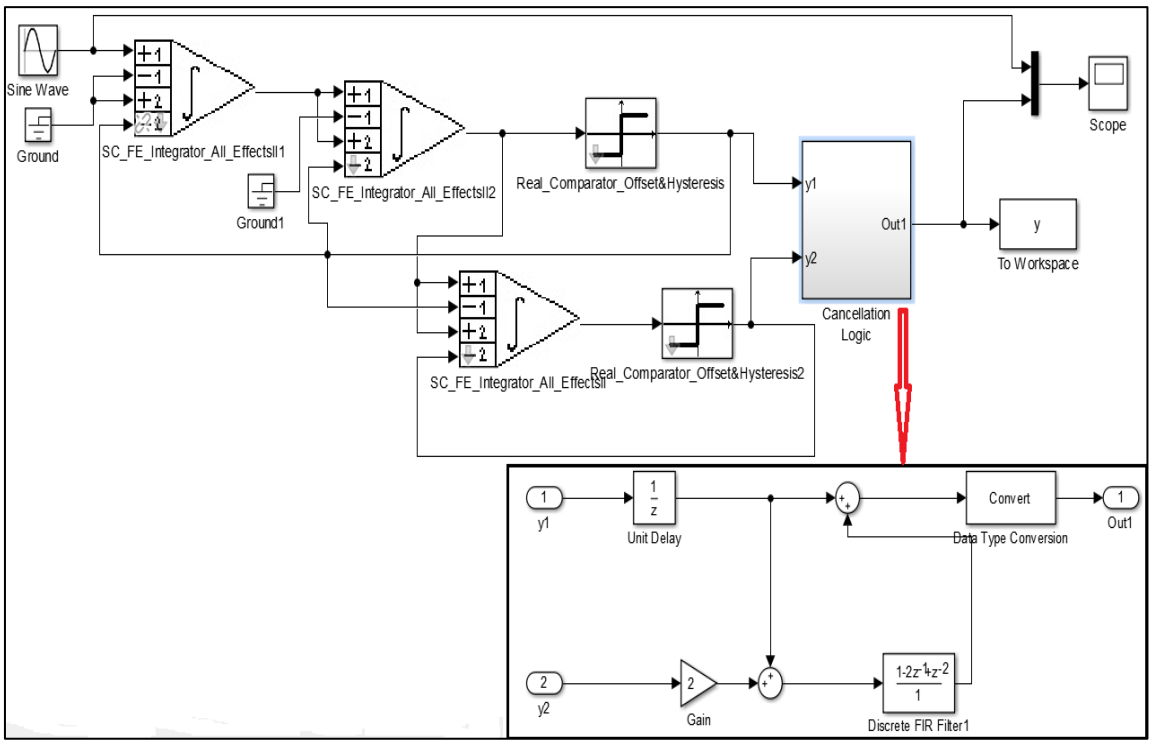
Figura 3.12. Diferentes sub-librerias incluidas en la libreria real de *Integrators*.

3.4.2 Configuración de parámetros del modelo

Para poder simular el M-ΣΔ de la Figura 3.13 b se necesitan los parámetros del modulador y los parámetros del modelo, estos pueden configurarse desde la ventana de comandos de MATLAB o pueden guardarse alternativamente en un archivo ***.m*** que se carga cuando es necesario.



a)



b)

Figura 3.13. Diagrama de bloques SIMSIDES del Modulador mostrado en la figura 3.7: (a) Construcción y edición del diagrama de bloques. (b) Diagrama completo de bloques del modulador en SIMSIDES.

Para no hacer muy agotador escribir todos los parámetros desde la ventana de comandos de Matlab se realizó el archivo ***.m*** llamado *Parameters*, donde se configuró todos los parámetros del modelo de la Figura 3.13 b, que incluye una breve descripción de los diferentes parámetros y variables incluidos, Figura 3.14.

Parameters.m

```
% SDM parameters:
% Sampling Frequency(fs), Input Frequency (fi), Sampling Time (Ts)
% OverSampling Ratio (OSR=M); Number of points (N)
fs=5.12e6; fi=5e3; Ts=1/fs; M=128; N=65536;
% Model parameters
kt=0.026*1.6e-19; % Boltzmann constant

% First Integrator's parameters
Cint1=24e-12; % integration capacitor For gain=1
Cs11=6e-12; % sampling capacitor (branch 1)
Cs21=6e-12; % sampling capacitor (branch 2)
innoise1=0; % rms value of the input equivalent noise
ao1=2.63e3; % open-loop OTA DC gain
gm1=4.5e-3; % transconductance
io1=0.977e-3; % maximum OTA output current
ron1=60; % sampling switch-on resistance

% Second- and Third- Integrators
Cint2=3e-12;
Cs12=1.5e-12;
Cs22=1.5e-12;
innoise2=0;
ao2=1.38e3;
gm2=0.87e-3;
io2=0.25e-3;
ron2=650;

% Common integrator parameters
temp=175; % temperature
osp=2.7; % output swing
cnl1=0; % capacitor first-order non-linear coef.
cnl2=25e-6; % capacitor second-order non-linear coef.
avnl1=0; % DC gain first-order non-linear coef.
avnl2=15e-2; % DC gain second-order non-linear coef.
avnl3=0; % DC gain third-order non-linear coef.
avnl4=0; % DC gain fourth-order non-linear coef.
cpar1=0.6e-12; % parasitic (opamp) input capacitance
cpar2=0.6e-12;
cload=2.28e-12; % opamp (intrinsic) load capacitance
% Comparators
vref=2; % DAC reference voltage
hys=30e-3; % comparator hysteresis
```

Figura 3.14. Archivo ***.m*** que incluye todos los parámetros del modelo para simular el M-ΣΔ de la Figura 3.13 b.

Para hacer más exacta la simulación se tuvo que incluir los valores de la Tabla 3.5 a todos los bloques del modulador, ya que incluye los valores de todos los parámetros de los bloques realizados en SIMSIDES como se muestra en la Figura 3.15, así como otros parámetros de bloques auxiliares (como los utilizados en los bloques Sine Wave y To Workspace) que se requieren durante la simulación. Además de estos parámetros del modelo, los parámetros de simulación deben configurarse para ejecutar una simulación. Para ello, vaya a

Simulation -> Model Configuration Parameters y defina los siguientes parámetros:

- *Simulation Time: Start time: 0.0; Stop time: (N-1) * Ts.*
- *Solver options: Type: Variable Step ; Max Step Size:* Auto

Cabe mencionar que los bloques de integración son identificados por orden para calcular correctamente las capacitancias de cargas equivalentes requeridas para la incompleta solución del error del modelo.

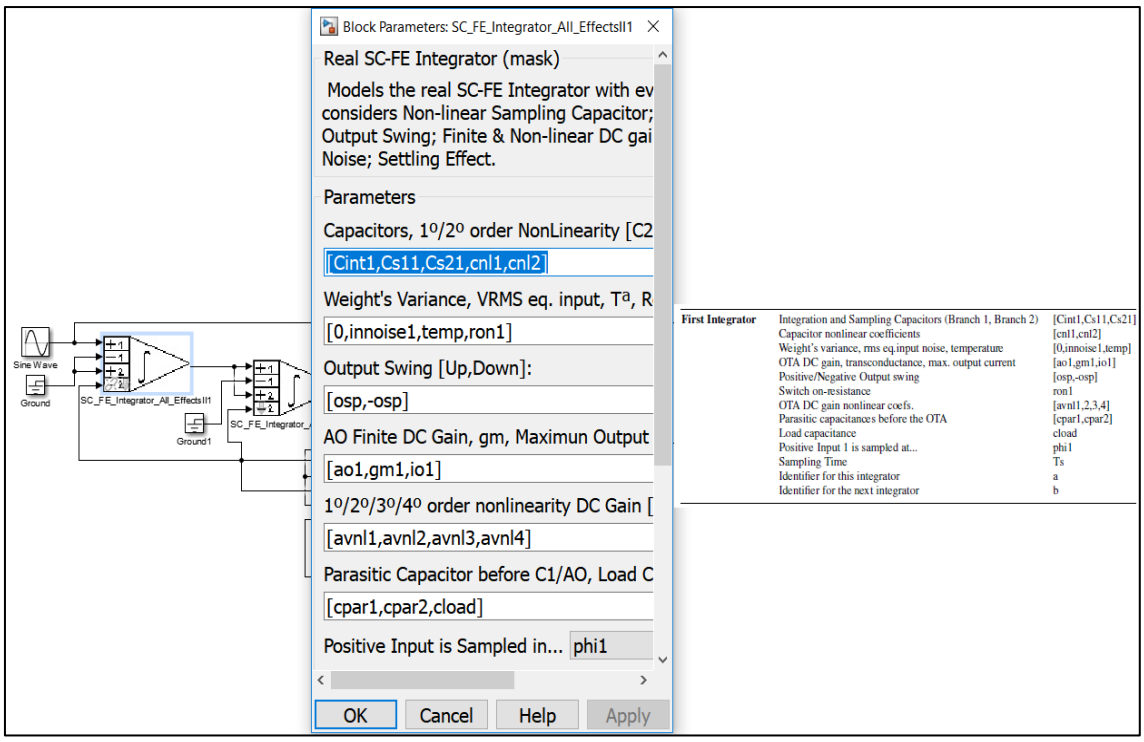


Figura 3.15. Configuración de parámetros de los bloques del M-ΣΔ para realizar la simulación.

3.4.3 Espectro de la salida de la simulación

El espectro de salida del M-ΣΔ se calcula en SIMSIDES realizando los siguientes pasos:

1. Configure los parámetros del modelo para usar el archivo ***.m*** mostrado en la Figura 3.14
2. Simule el modulador de la Figura 3.13 b desde el menú *Simulation -> Run*.
3. Cuando la simulación termine, ir a la interfaz de SIMSIDES *Analysis -> Node Spectrum Analysis*.
4. Define los parámetros requeridos en el menú. En este ejemplo la frecuencia de muestreo es definido como *fs* la Función de la ventana de

Kaiser es usado con un número de puntos *N* y *Beta* =20 como se muestra en la Figura 3.16.

5. Dar clic en *Compute* y después seleccionamos el nombre de la salida en este ejemplo *y*, damos nuevamente clic en *Compute* y finalmente *Plot*. La salida del espectro se muestra en el apartado de resultados.

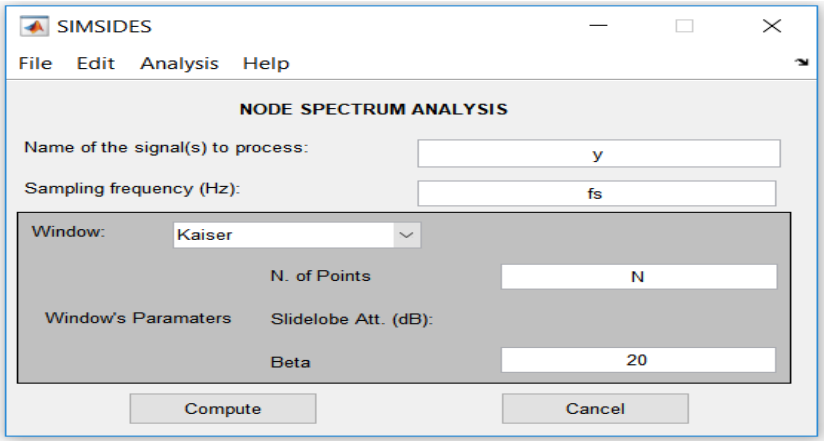


Figura 3.16. Configuración de parámetros en SIMSIDES para el M-ΣΔ para poder realizar la simulación.

Tabla 3.5. Parámetros del modelo de los bloques de construcción para simular el M-ΣΔ [27].

Building Block	Parameter Description	Value/Variable
Input Sine Wave	Sine Type	Time based
	Amplitude	0.5
	Bias	0
	Frequency (rad/s)	2*pi*fi
	Phase (rad)	0
	Sample time	0
	Interpret vector parameters	Selected
First Integrator	Integration and Sampling Capacitors (Branch 1, Branch 2)	[Cint1,Cs11,Cs21]
	Capacitor nonlinear coefficients	[cnl1,cnl2]
	Weight's variance, rms eq.input noise, temperature	[0,innoise1,temp]
	OTA DC gain, transconductance, max. output current	[ao1,gm1,io1]
	Positive/Negative Output swing	[osp,-osp]
	Switch on-resistance	ron1
	OTA DC gain nonlinear coefs.	[avnl1,2,3,4]
	Parasitic capacitances before the OTA	[cpar1,cpar2]
	Load capacitance	cload
	Positive Input 1 is sampled at...	phi1
	Sampling Time	Ts
	Identifier for this integrator	a
	Identifier for the next integrator	b
Second, Third Integrators	Integration and Sampling Capacitors (Branch 1, Branch 2)	[Cint2,Cs12,Cs22]
	Capacitor nonlinear coefficients	[cnl1,cnl2]
	Weight's variance, rms eq.input noise, temperature	[0,innoise2,temp]
	OTA DC gain, transconductance, max. output current	[ao2,gm2,io2]
	Positive/Negative Output swing	[osp,-osp]
	Switch on-resistance	ron2
	OTA DC gain nonlinear coefs.	[avnl1,2,3,4]
	Parasitic capacitances before the OTA	[cpar1,cpar2]
	Load capacitance	cload
	Positive Input 1 is sampled at...	phi1
	Sampling Time	Ts
	Identifier for this integrator (second integrator)	b
	Identifier for this integrator (third integrator)	c
	Identifier for the next integrator	c
Comparators	Vhigh, Vlow	[vref, -vref]
	Offset, Hysteresis	[0,hys]
	Phase ON	phi1
	Sampling Time	Ts
	Identifier for this quantizer	quant1
To Workspace (y)	Variable name	y
	Limit data points to last	N
	Decimation	1
	Sample Time	Ts
	Save format	Array

3.5 Exportar los modelos a FPAA

Para realizar esta etapa se utilizó la tarjeta Anadigm AN231E04. Para implementar el modulador de la Figura 1 en la tarjeta FPAA Anadigm AN231E04 se utilizó el kit de desarrollo AN231K04-DVLP3 que es una plataforma muy amigable de utilizar para la implementación y pruebas de circuitos analógicos. Contiene una interfaz PC serial para poder descargar los circuitos diseñados en el software AnadimgDesigner2 (AD2). De la misma forma, se utilizó la plataforma NI Elvis II para generar la señal sinusoidal.

Después de haber instalado con éxito AD2 y CP210x, se llevan a cabo las configuraciones en la tarjeta FPAA, como se muestra en la Figura 3.17 para poder bajar el diseño del modulador realizado en AD2 a la tarjeta. En la Figura 3.18 se presenta la conexión de un diferenciador de voltaje en la plataforma NI Elvis II, para generar la señal de entrada sinusoidal.

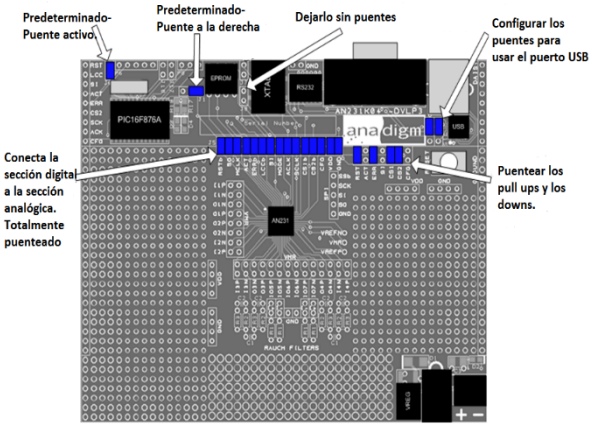


Figura 3.17 Configuración de la tarjeta FPAA AN231E04 con puentes.

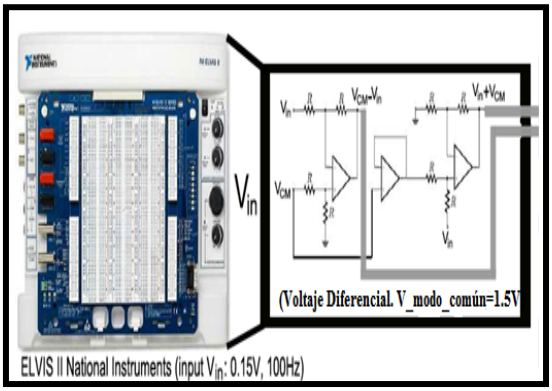


Figura 3.18 Acoplamiento de un diferenciador de voltaje en la plataforma NI Elvis II.

Posteriormente, se realizó la conexión (Figura 3.19) de todo el Sistema para *bajar* el modulador ΣΔ a la tarjeta FPAA. Donde la interfaz NI Elvis nos ayuda a la alimentación de la tarjeta con un voltaje de 9V, además de manipular el voltaje y la frecuencia de la señal de entrada como se puede observar en la Figura 3.20. Por último, se diseñó el modulador ΣΔ en Anadimg Designer2 como se muestra en la Figura 3.21, el cual se exportó a la tarjeta FPAA.

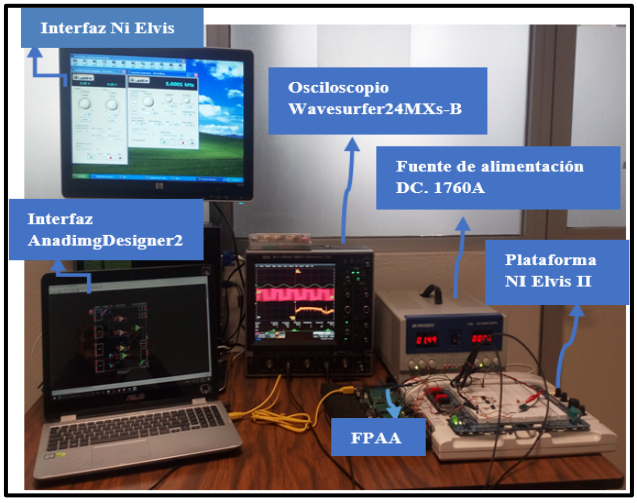


Figura 3.19 Configuración y conexión del Sistema.

Figura 3.20 Interfaz NI Elvis II.

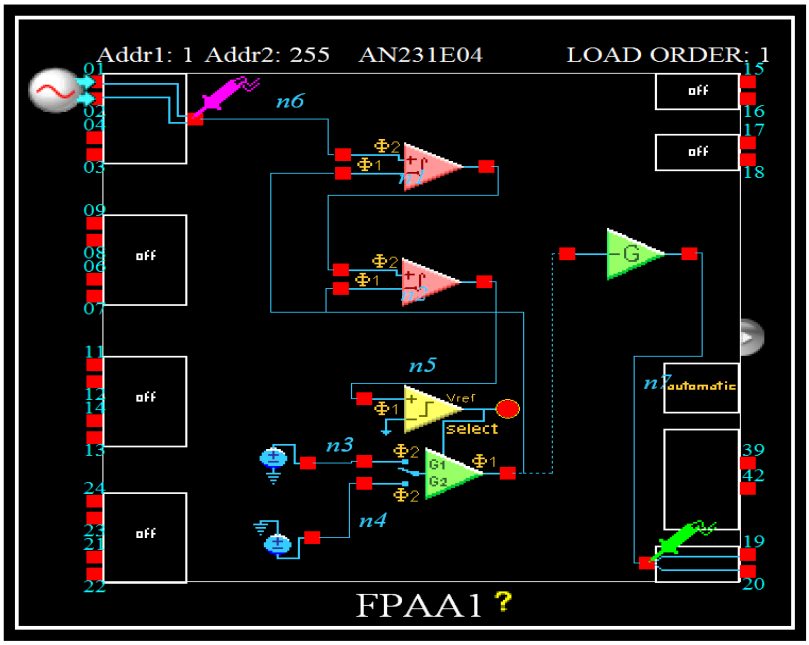


Figura 3.21 Modulador ΣΔ de segundo orden en AD2.

Capítulo 4 Resultados y discusión

En este capítulo se describen los resultados obtenidos basados en la metodología desarrollada en el capítulo 3.

4.1 Modelado comportamental

Para poder realizar el convertidor ΣΔ analógico-digital se empezó con el diseño del modulador ΣΔ en cascada de segundo y tercer orden como se muestra en las Figura 4.1 y 4.2, haciendo uso de los siguientes bloques: Integradores, comparadores, entre otros. Todo esto con la herramienta SIMSIDES. Teniendo como resultado los moduladores mostrados en las Figuras 4.8 y 4.9.

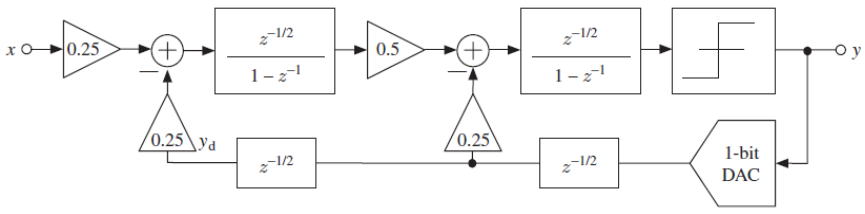


Figura 4.1 Diagrama de bloques del modulador ΣΔ TD de 2° orden

En la Figura 4.1 se pueden observar un modulador ΣΔ con dos integradores lo cual se consigue aumentar hasta dos el orden de la función de transferencia del ruido de cuantización. Por otra parte esto hace que la

densidad espectral de potencia del ruido de cuantización disminuya mucho en las zonas de bajas frecuencias al precio de un pequeño aumento en las zonas de altas frecuencias.

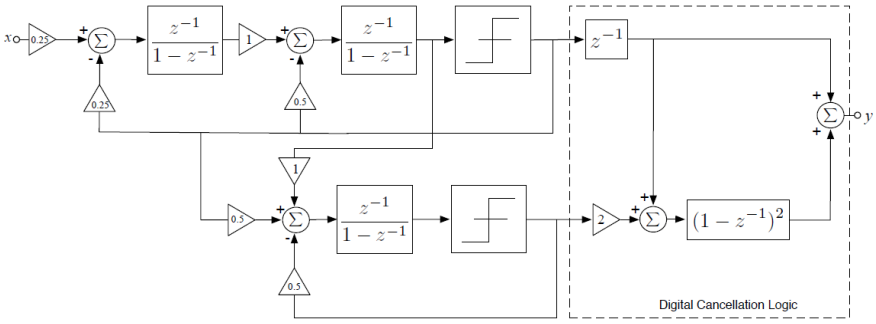


Figura 4.2 Diagrama de bloques del modulador ΣΔ TD de 3° orden en cascada

En la Figura 4.2 se observa una conexión en cascada de moduladores de bajo orden (1 y 2), en este modulador se tiene tres integradores por lo que se convierte en un modulador de tercer orden. El ruido de cuantización generado en una etapa es remodulado por las siguientes y posteriormente cancelado en el dominio digital.

4.2 Programación

En la herramienta SIMSIDES la mayoría de los bloques fueron programados para la parte de las no idealidades, en el caso de los integradores de tiempo discreto las *S-Funtion* fueron programados en el lenguaje de MATLAB. Teniendo el modelo completo del modulador se programó cada bloque con los valores correspondientes a la arquitectura (Figura 4.5), donde el valor de cada constante se guardó en un archivo con extención **.m** llamado *Parameters.m* como se observa en la Figura 4.6.

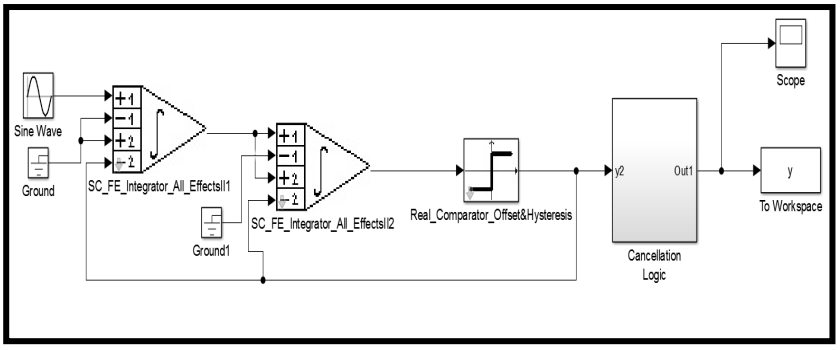


Figura 4.3 Diagrama de bloques de SIMSIDES del modulador ΣΔ de segundo orden.

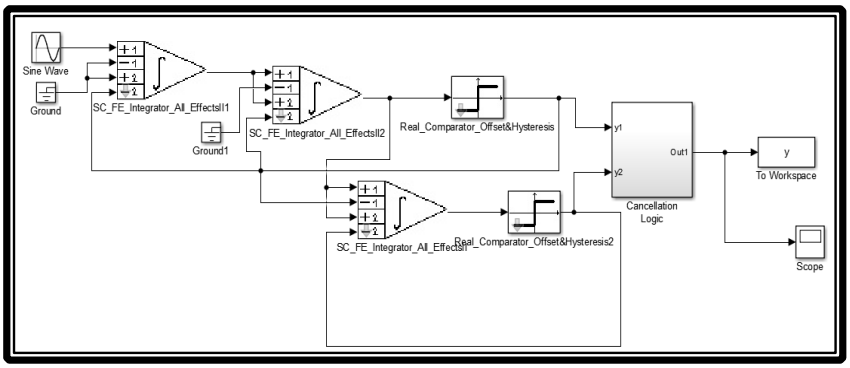
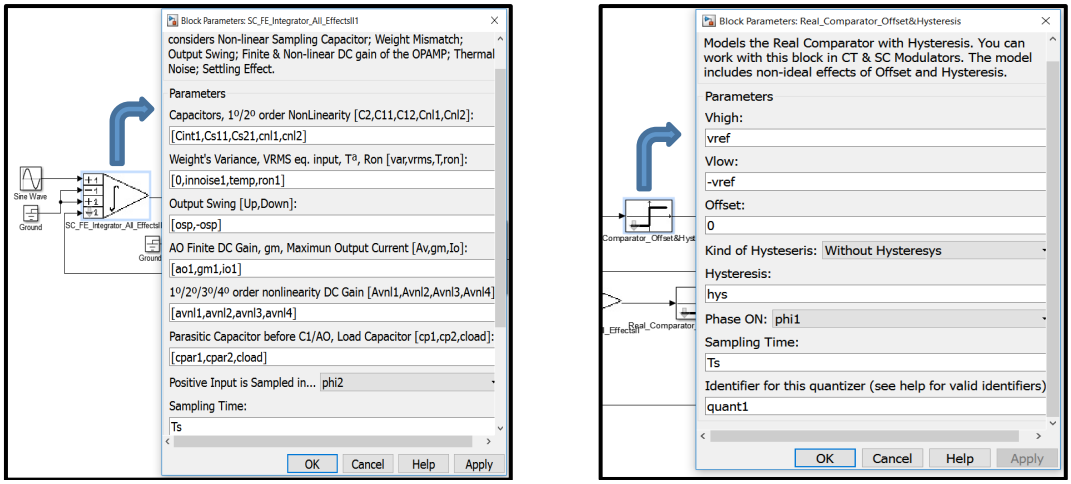


Figura 4.4 El diagrama de bloques de SIMSIDES del Modulador de tercer orden de la Figura 4.2

Para el integrador que se utilizó en el diseño de los moduladores de las Figuras 4.3 y 4.4 contienen dos *S-Function* (intfestc2ramascompleto y salidafe) y bloques necesarios para su funcionamiento. El comparador contiene un bloque (cuantrealhist).



a) b)

Figura 4.5 Programación de parámetros a cada bloque del modulador: a) Integrador,b) Comparador.

Para poder utilizar el integrador y comparador en cada modulador se programó cada bloque con las constantes especificas para su funcionamiento (Figura 4.5), el valor de las constamtes (Frecuencia de muestreo, frecuencia de entrada, timpo de muestreo, sobremuestreo, capacitores, ganancias, etc.) son guardadas en un archivo ***.m*** (Figura 4.6), ya que nos facilita cambiar los valores de las constantes si así se requiere.

```
% SDM parameters:
% Sampling Frequency(fs), Input Frequency (fi), Sampling Time (Ts)
% OverSampling Ratio (OSR=M); Number of points (N)
fs=5.12e6; fi=5e3; Ts=1/fs; M=128; N=65536;
% Model parameters
kt=0.026*1.6e-19; % Boltzmann constant

% First Integrator's parameters
Cint1=24e-12; % integration capacitor For gain=1
Cs11=6e-12; % sampling capacitor (branch 1)
Cs21=6e-12; % sampling capacitor (branch 2)
innoise1=0; % rms value of the input equivalent noise
ao1=2.63e3; % open-loop OTA DC gain
gm1=4.5e-3; % transconductance
io1=0.977e-3; % maximum OTA output current
ron1=60; % sampling switch-on resistance

% Second- and Third- Integrators
Cint2=3e-12;
Cs12=1.5e-12;
Cs22=1.5e-12;
innoise2=0;
ao2=1.38e3;
gm2=0.87e-3;
io2=0.25e-3;
ron2=650;

% Common integrator parameters
temp=175; % temperature
osp=2.7; % output swing
cnl1=0; % capacitor first-order non-linear coef.
cnl2=25e-6; % capacitor second-order non-linear coef.
avnl1=0; % DC gain first-order non-linear coef.
avnl2=15e-2; % DC gain second-order non-linear coef.
avnl3=0; % DC gain third-order non-linear coef.
avnl4=0; % DC gain fourth-order non-linear coef.
cpar1=0.6e-12; % parasitic (opamp) input capacitance
cpar2=0.6e-12;
cload=2.28e-12; % opamp (intrinsic) load capacitance
% Comparators
vref=2; % DAC reference voltage
hys=30e-3; % comparator hysteresis
```

Figura 4.6 Archivo *.m* de valores de parámetros del modulador ΣΔ.

4.3 Pruebas (Simulación en Matlab)

En las Figuras 4.7 y 4.8 se muestran los espectros de salida del modulador de segundo y tercer orden respectivamente. Las simulaciones se realizaron con los mismos valores de los parámetros del archivo *.m*. Para poder obtener las salidas correctas se ajustó el módulo de cancelación lógica, dando como respuesta los espectros mostrados en la Figura 4.9.

Para la simulación de los moduladores de segundo y tercer orden se utilizó el módulo de cancelación lógica. Para el M-ΣΔ de segundo orden se utilizó en el módulo de cancelación una ganancia de 0.5 y para el M-ΣΔ de tercer orden se utilizó una ganancia de 2, esto nos dio como resultado que la magnitud del espectro de la Figura 4.7 sea mayor que la magnitud del espectro de la Figura 4.8, sin embargo con el módulo de cancelación podemos ajustar la ganancia (0.5-2) para que la magnitud del espectro del modulador de segundo orden sea aproximadamente el mismo que el de tercer orden.

Por otra parte, el modulador de tercer orden presenta mejores patrones de ruido en presencia de señales estáticas, además de que la aparición de espúreos en la banda de la señal disminuye a medida que aumenta el orden del modulador como se muestra en la Figura 4.10.

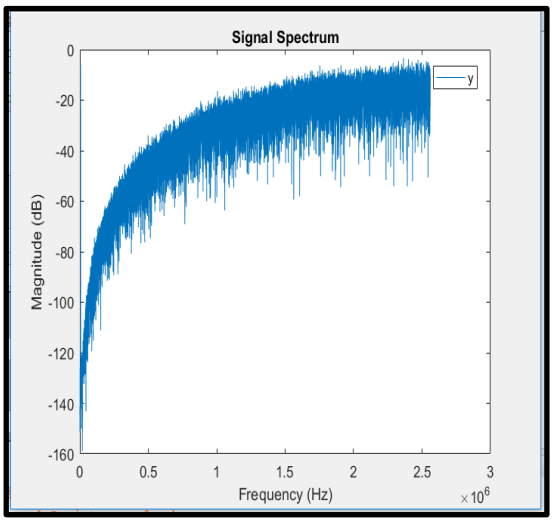


Figura 4.7 Espectro de salida del modulador ΣΔ de 3° orden.

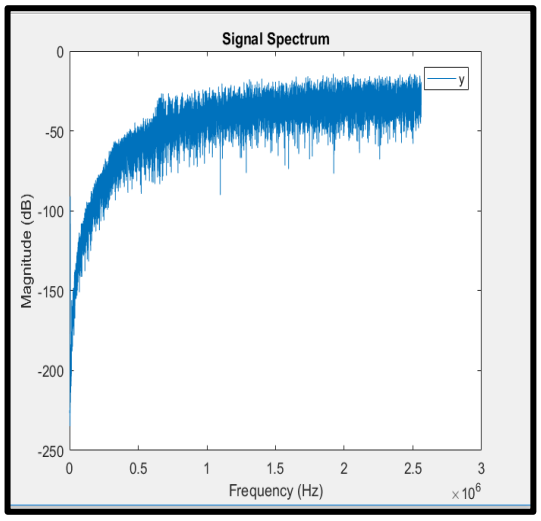


Figura 4.8 Espectro de salida del modulador ΣΔ de 2° orden.

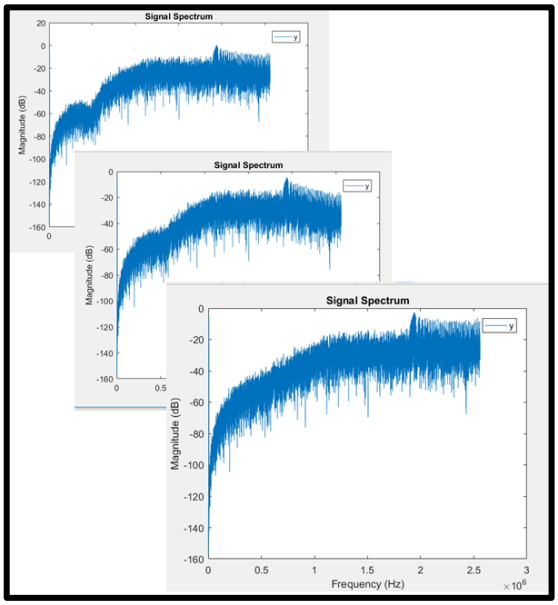


Figura 4.9 Respuesta del modulador ΣΔ de 2° orden ajustando el módulo de cancelación lógica

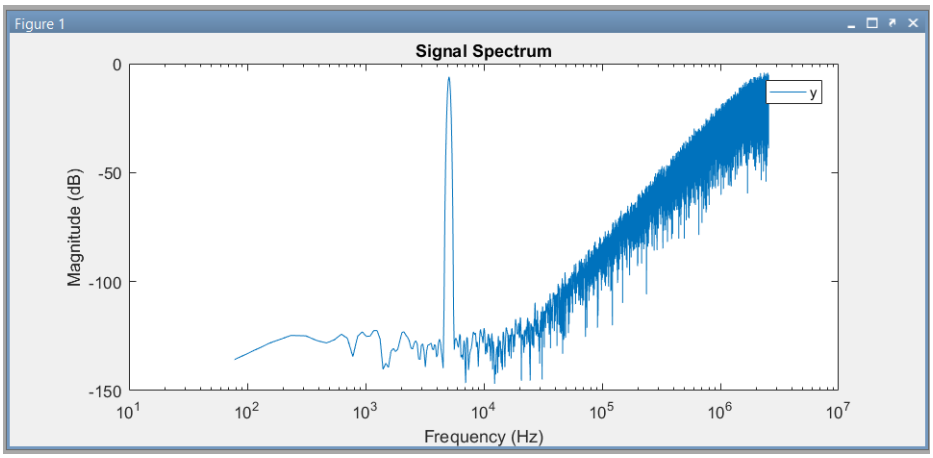


Figura 4.10. Espectro de salida del M-ΣΔ en escala logarítmica.

4.4 Exportar los modelos a FPAA

En la Figura 4.11 Podemos observar la señal sinusoidal de $1V_{pp}$ (color verde), a una frecuencia de 5kHz, con una frecuencia de reloj de 4MHz para alimentar al modulador, dando una señal modulada a la salida (color rosa) y el espectro correspondiente a la señal de salida (color amarillo), todo esto en la tarjeta FPAA.

Se realizaron simulaciones de diferentes valores de amplitud (0.3-1.8) para obtener el mejor resultado similar al resultado de la simulación en Matlab (Anexo C). Al aumentar la amplitud la energía se va dirigiendo hacia afuera de la banda de la señal, donde posteriormente será eliminada por un filtro (Decimador).

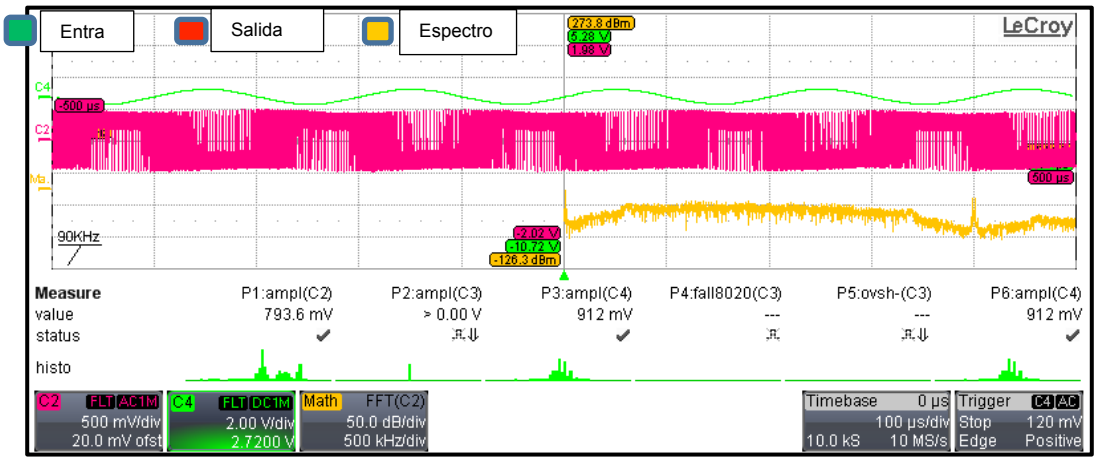


Figura 4.11 Señal de entrada, salida y espectro del modulador ΣΔ de segundo orden.

En las Figuras 4.12 y 4.13 se muestran los dos espectros del modulador ΣΔ siendo estas la respuesta de la tarjeta FPAA. En la primera figura se observa el comportamiento de las frecuencias altas que se trasladan hacia la

salida y en la segunda figura se observa el pico de la amplitud del comienzo de la señal de salida.

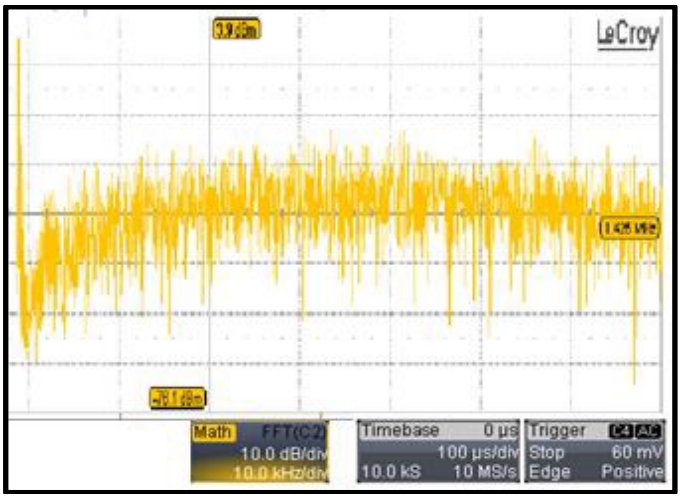


Figura 4.12 Espectros de salida del modulador ΣΔ de segundo orden.

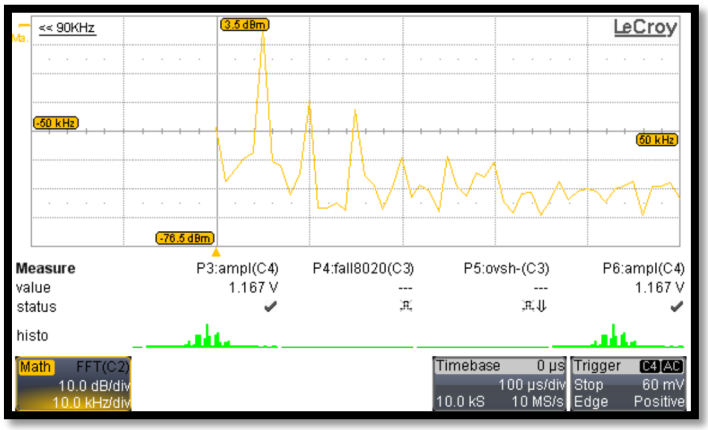


Figura 4.13 Densidad de potencia espectral de la salida del modulador sigma-delta, implementado para una señal de entrada de 5 kHz en FPAA.

Capítulo 5 Conclusiones y perspectivas

5.1 Conclusiones

Con los resultados obtenidos se puede observar y corroborar que es posible utilizar Matlab para diseñar sistemas electrónicos mediante *toolboxes*, siento específicos los moduladores-ΣΔ e implementarlos en sistemas embebidos. Es cierto que se cuenta con sistemas electrónicos que se pueden implementar desde Simulik pero es con hardware muy dedicado como lo son los microcontroladores y FPGAs, sin embargo en este proyecto se hizo uso de la tarjeta FPAA que es algo menos común.

Las principales ventajas de SIMSIDES en comparación con otros simuladores comportamentales como Verilog-A es la buena relación entre el tiempo de simulación y la precisión, además de tener la opción de implementarlo en un circuito integrado, resaltando que los modelos de comportamiento de estos componentes básicos tienen en cuenta los mecanismos de error más críticos de las diferentes técnicas de circuitos, incluidos los circuitos de condensador conmutado (SC), de corriente conmutada (SI) y de tiempo continuo (CT), así mismo, los bloques de las demás topologías de moduladores pueden ser utilizados por otros sistemas electrónicos.

Los resultados experimentales obtenidos en la simulación del modulador ΣΔ de segundo orden en Simulink/Matlab y los obtenidos de la tarjeta FPAA, podemos concluir que las tarjetas reconfigurales son una gran opción de usar si no se cuenta con los recursos y el tiempo para poder bajarlo a un Circuito Integrado. Ya que como se describe en éste proyecto bajar un diseño a la tarjeta FPAA no resulta tan costoso.

5.2 Perspectivas

Bajar el modulador-ΣΔ a otros hardware reconfigurables como el FPGA o a un microcontrolador.

Realizar la parte del Decimador para obtener el convertidor analógico-digital ΣΔ.

Referencias bibliográficas

[1] L. L. a. G. M. Louis Scheffer, EDA for IC Implementation, Circuit Design, and Process Technology, Boca Raton, FL: Taylor & Francis Group, 2006, pp. 1-5.

[2] I. H. Juan, Conversores Análogo-Digital y Digital-Análogo: Conceptos Básicos, Bogotá: Univerdidad Nacional Abierta y a Distancia, 2015, pp. 1-3.

[3] B. Razvi, Principles of Data Conversion System Design, Piscataway: IEEE PRESS, 1995, pp. 148-74.

[4] A. G. Raphael da Costa Neves, F ast Prototyping of ΣΔ Modulators Using Reconfigurable Analog Arrays, Brasil: BDBComp, 2011.

[5] D. a. B. V. R. Reddy, «Field programmable analog array: A boon for analog world,» *IEEE Xplore Digital Library,* pp. 2975-2980, 2016.

[6] EUROPRACTICE, [En línea]. Available: http://www.europractice-ic.com/docs/160401_MPW2016-general-v7.pdf. [Último acceso: 07 06 2016].

[7] EUROPRACTICE, [En línea]. Available: http://www.europractice-ic.com/docs/2016%20EP%20wall%20plan.pdf. [Último acceso: 07 06 2016].

[8] S. Ian, Ingeniería del software, vol. Séptima edición, Madrid, España.: PEARSON EDUCACIÓN, S.A., 2005, pp. 3-5,94-95.

[9] S. P. Roger, Ingenieria del Software, vol. Séptima edición , México, D.F: The McGraw-Hill, 2010, pp. 10-12.

[10] O. BEUCHER, M. WEEKS., Introduction to MATLAB & SIMULINK, Hingham: M.A., 2008, pp. 1-3.

[11] G. C. T. Richard Schreier, Understanding Delta-Sigma Data Converters, Oregon: IEEE Press, 2004, pp. 1-5.

[12] M. S. a. W. S. Libin Yao, Low-Power Low-Voltage Sigma-Delta Modulators in Nanometer CMOS, Dordrecht, The Netherlands: Springer, 2006, p. 25.

[13] M. O. a. F. Gerfers, Continuous-Time Sigma-Delta A/D Conversion. Fundamentals, Performance Limits, Netherlands: Springer, 2006, pp. 19-22.

[14] S. LOGAN, «Which A/D converter is right for my application?,» *Electronic products,* 2015.

[15] Profesores UNAM, «Tecnologías para el procesamiento digital de señales (notas de clase),» 23 Junio 2014. [En línea]. Available: http://profesores.fi-b.unam.mx/maixx/Biblioteca/Librero_Telecom/Libro_TecnologiasProcSeniales/TecnologiasProcSeniales_02_04_01.pdf. [Último acceso: 5 Junio 2016].

[16] J. M. d. l. R. a. R. d. Rio, Cmos Sigma-Delta Converters. Practical Desing Guide, Chennai, India: John Wiley & Sons, Ltd., 2013, pp. 5-7, 9-10,15-48, 121-128.

[17] B. Razavi, Principles of Data Conversion System Design, Piscataway: IEEE PRESS, 1995, pp. 45-47.

[18] D. E. J. Peralías, «INAOE,» 21-23 09 2011. [En línea]. Available: http://www-elec.inaoep.mx/seminario2011/castour-edsdlp/Peralias_INAOE_sep11.pdf. [Último acceso: 05 07 2016].

[19] H. Zumbahlen, Linear Circuit Design Handbook, Burlington, USA: Analog Devices, 2008, pp. 6.138-141.

[20] M. A. S. a. E. A. J. M. v. Tuijl, Power Trade-Offs and Low-Power in Analog CMOS ICs, Boston: Kluwer Academic Publishers, 2010, pp. 36,37.

[21] D. R. D. a. P. G. Gulak, «Design Approaches to Field-Programmable Analog Integrated Circuits,» *Kluwer Academic Publishers,* pp. 8-34, 1997.

[22] Anadigm, «The Anadigm Product Range,» [En línea]. Available: http://www.anadigm.com/products.asp. [Último acceso: 20 06 2016].

[23] Anadigm, «AN231E04 Datasheet Rev. 1.2,» 2014. [En línea]. Available: http://www.anadigm.com/_doc/DS231000-U001.pdf. [Último acceso: 27 06 2016].

[24] T. S. H. a. C. M. T. a. P. H. a. D. V. Anderson, «Developing large-scale field-programmable analog arrays for rapid prototyping,» *International Journal of Embedded Systems,* vol. 1, nº 3/4, pp. 179-

192, 2005.

[25] Teodoro Álamo Cantarero, Daniel Limón Marruedo, Manuel Gil Ortega Linares, Manuel Ruiz Arahal y Guillermo Heredia Benot, Introducción al Simulink, Modelado y simulación de sistemas dinámicos, Sevilla: Departamento de Ingeniería de Sistemas y Automática , 2000, pp. 15,26.

[26] Timon Bruckner, Matthias Lorenz, Christoph Zorn, Joachim Becker, Wolfgang Mathis and Maurits Ortmanns, «Hardware-Accelerated Simulation Environment for CT Sigma–Delta Modulators Using an FPGA,» *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS,* vol. 59, nº 8, pp. 471-475, August 2012.

[27] J. M. d. l. R. a. R. d. Río, *SIMSIDES User Guide,* Sevilla, 2013.

[28] J. M. d. l. Rosa, «Using Behavioral Modeling and Simulation for Learning Communication Circuits and Systems,» de *Global Engineering Education Conference (EDUCON)*, Marrakech, 2012.

[29] P.J. Obeso-Rodelo, J.A. Galaviz-Aguilar, J.A. Sillas, J.M. Jiménez, J.C. Núñez Pérez y E. Tlelo Cuautle., Metodología de diseño e implementación de un oscilador caótico de Chua en un FPGA., Chihuahua: CITEDI-IPN, INAOE., 2014.

[30] K. W. a. R. S. a. J. Marciniak, «A Competitive WTA ANN for Classification a Calorific Value of a Coal Fuel in Combustion Chambers Using FPAA,» *International Conference on Signals and Electronic Systems (ICSES),* pp. 5-7, 2016.

[31] a. S. K. Jennifer Hasler and Aishwarya Natarajan and Sahil Shah, «SoC FPAA Immersed Junior Level Circuits Course,» *IEEE International Conference on Microelectronic Systems Education,* pp. 7-10, 2017.

[32] P. P. a. P. D. Shashant Jaykar, «Modeling of Sigma-Delta Modulator Non-Idealities in MATLAB/SIMULINK,» de *International Conference on Communication Systems and Network Technologies*, 2011.

[33] V. K. J. H. M. PAVLIK, «Design of the 12-bit Delta-Sigma Modulator using SC Technique for Vibration Sensor Output Processing,» *RADIOENGINEERING,* vol. 21, nº 1, pp. 246-251, 2012.

[34] J. M. d. l. R. a. R. d. Río, CMOS SIGMA-DELTA CONVERTERS. PRACTICAL DESIGN GUIDE, Chennai, India: John Wiley & Sons,

Ltd., 2013, pp. 5-7, 9-10,12-48, 136,137,160-171.

[35] C. I. J. Sánchez, *Evaluación e implementación de algoritmos de adquisición de una señal GPS, en un dispositivo reconfigurable FPGA,* México D.F.: Instituto Politécnico Nacional, 2009, pp. 23-24.

[36] ALTERA, [En línea]. Available: https://www.altera.com/products/boards_and_kits/dev-kits/altera/kit-cyclone-iv-gx.html. [Último acceso: 08 06 2016].

[37] ALTERA, [En línea]. Available: https://www.altera.com/products/boards_and_kits/daughter-cards.html. [Último acceso: 08 06 2016].

[38] «ANALOG-DIGITAL CONVERSION,» de *FUNDAMENTALS OF SAMPLED DATA SYSTEMS*, Massachusetts, One technology way, 1986, p. 3.

[39] J. Gerardo García-Sánchez and José M. de la Rosa, «Efficient hybrid continuous-time/discrete-time cascade ΣΔ modulators for wideband applications,» *ELSEVIER,* p. 1234–1246, 2014.

Anexos

Anexo A

Código de los bloques programados en lenguaje C y Matlab para modelos comportamentales.

El modelo comportamental del integrador en tiempo discreto (ecuaciones 3.3 y 3.4), se muestra en A.1. Donde C1 y C2 son, respectivamente, la capacitancia de muestreo (Cs) y la capacidad de integración (Ci), y *count* es un parámetro que determina la fase de reloj, *count* =0 corresponde a la fase de muestreo y *count* =1 a la fase de integración. La señal de entrada es modelada por una matriz formada por dos vectores u (1) y u (2), mientras que la salida se almacena en una variable denominada *y*, modelando *yold* la muestra de salida almacenada en la fase de reloj anterior y siendo *ytemp* una variable de estado temporal utilizada hasta que se alcanza la convergencia [16].

```
function y = intfeavnl(u,AV,AVNL1,AVNL2,AVNL3,AVNL4,C1,C2,PHI)
persistent VC1 VC2 VC1OLD count yold
if (isempty(VC1))
    VC1=0;
    VC2=0;
    VC1OLD=0;
    count=0;
    yold=0;
end
     if (PHI==1)
          if (count==0)
              VC1 =u(1);
              VC2 =yold;
              y=yold;
              count=count+1;
          else
              VC1OLD = VC1;
              VC1 = u(2);
              AVNEW = AV;
              ITER = 0;
              AVVAR=1;
              while (AVVAR>0.01&ITER<50),
                  ytemp = ((C1/C2)*(VC1OLD-VC1)*(AVNEW/(1+AVNEW+(C1/C2)))) + (VC2*((AVNEW+1)/(1+AVNEW+(C1/C2))));
                  AVOLD = AVNEW;
                  VO2 = ytemp^2;
                  VO3 = ytemp^3;
                  VO4 = ytemp^4;
                  AVNEW = AV*(1+(AVNL1*ytemp)+(AVNL2*VO2)+(AVNL3*VO3)+(AVNL4*VO4));
                  AVVAR = abs((AVOLD-AVNEW)/AVNEW);
                  ITER = ITER + 1;
              end
              y = ytemp;
              yold=y;
              count=0;
          end
    elseif (count==0)
              VC1OLD = VC1;
              VC1 = u(2);
              AVNEW = AV;
              ITER = 0;
              AVVAR=1;
              while (AVVAR>0.01&ITER<50),
                  ytemp = ((C1/C2)*(VC1OLD-VC1)*(AVNEW/(1+AVNEW+(C1/C2)))) + (VC2*((AVNEW+1)/(1+AVNEW+(C1/C2))));
                  AVOLD = AVNEW;
                  VO2 = ytemp^2;
                  VO3 = ytemp^3;
                  VO4 = ytemp^4;
                  AVNEW = AV*(1+(AVNL1*ytemp)+(AVNL2*VO2)+(AVNL3*VO3)+(AVNL4*VO4));
                  AVVAR = abs((AVOLD-AVNEW)/AVNEW);
                  ITER = ITER + 1;
              end
              y = ytemp;
              yold=y;
              count=count+1;
          else
              VC1 = u(1);
              VC2 = yold;
              y=yold;
              count=0;
          end
```

A.1 Código en Matlab del modelo comportamental del integrador SC FE de la Figura 3.4a incluyendo el efecto de la no linealidad de la ganancia del amplificador [19].

El modelo de un integrador Gm-C (Figura 3.4b) también puede implementarse como la cascada de dos bloques de construcción. El bloque de transductor S-función (A.2) incluye el ruido referido a la entrada, la tensión de saturación en los nodos de entrada y de salida, y una transconductancia que es una función no lineal de la tensión de entrada.

```
#define S_FUNCTION_NAME pl_integrator
#define  S_FUNCTION_LEVEL 2
#include "simstruc.h"
#include <math.h>
#define gm(S) ssGetSFcnParam(S,0) /* Nominal transconductance */
#define C(S)  ssGetSFcnParam(S,1) /* Nominal capacitance */
#define et(S) ssGetSFcnParam(S,2) /* Time constant error (%) */
#define Av(S) ssGetSFcnParam(S,3) /* Finite DC gain */
#define ub(S) ssGetSFcnParam(S,4) /* Output swing limitation (max) */
#define lb(S) ssGetSFcnParam(S,5) /* Output swing limitation (min) */
#define iub(S) ssGetSFcnParam(S,6)/* Input swing limitation (max) */
#define ilb(S) ssGetSFcnParam(S,7)/* Input swing limitation (min) */
#define gm1(S) ssGetSFcnParam(S,8)/* First-order transconductance non-linearity */
#define gm2(S) ssGetSFcnParam(S,9)/* Second-order transconductance non-linearity */
#define selector_param(S) ssGetSFcnParam(S,10)
#define N_PARAMS 11

static void mdlInitializeSizes (SimStruct *S)
{...}

/* Continuous-time nature of the circuit is defined here */

static void mdlInitializeSampleTimes(SimStruct *S)
{
        ssSetSampleTime( S, 0, CONTINUOUS_SAMPLE_TIME);
        ssSetOffsetTime ( S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS
static void mdlInitializeConditions (SimStruct *S)
{
        real_T *x0 = ssGetContStates(S) ;
        x0[0] = 0; /* State variable */
}

static void mdlOutputs (SimStruct *S, int_T tid)
{
        real_T  *y      = ssGetOutputPortRealSignal (S, 0);
        real_T  *x      = ssGetContStates (S);
        real_T ub = *mxGetPr(ub(S));
    real_T lb = *mxGetPr(lb(S));
```

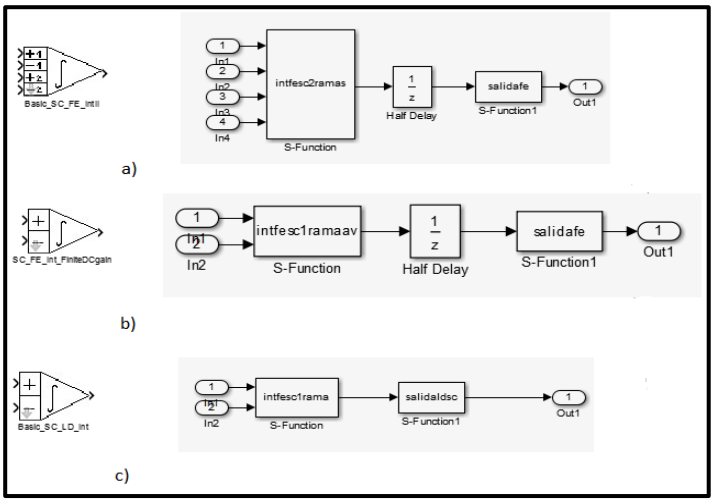
```
        /*Checking output saturation*/

    if(x[0]<=lb)
        {x[0]=lb;}
    if(x[0]>=ub)
        {x[0]=ub;}

        y[0] = x[0] ;
}

#define MDL_DERIVATIVES

static void mdlDerivatives(SimStruct *S)
{
    real_T *dx  = ssGetdX(S);
        real_T *x = ssGetContStates (S);
        real_T ub = *mxGetPr(ub(S));
    real_T lb = *mxGetPr(lb(S));
        real_T iub = *mxGetPr(iub(S));
    real_T ilb = *mxGetPr(ilb(S));
        real_T gm1 = *mxGetPr(gm1(S));
    real_T gm2 = *mxGetPr(gm2(S));
        real_T C = *mxGetPr(C(S));
    real_T gm = *mxGetPr(gm(S));
        real_T et = *mxGetPr(et(S));
    real_T Av = *mxGetPr(Av(S));
        real_T  selector = *mxGetPr(selector_param(S));
    InputRealPtrsType uPtrs0 = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType uPtrs1 = ssGetInputPortRealSignalPtrs(S,1);
    InputRealPtrsType uPtrs2 = ssGetInputPortRealSignalPtrs(S,2);
    real_T vin = *uPtrs0[0]; /* Input voltage */
    real_T iin = *uPtrs1[0]; /* Input current (from the modulator feedback DAC */
    real_T noise = *uPtrs2[0];/* Input noise source */
    real_T input;
    real_T go; /* Output conductance */
    C=C*(1+et); /* Integration capacitance modification due to time-constant error */
    go=gm/Av;   /* Finite DC gain: Av=gm/go */

        if (selector ==1) /* Input voltage without noise contribution */
                {input=vin;}
        else
                {input=vin+noise;} /* Input voltage considering noise contribution */

    input=input+gm1*pow(input,2)+gm2*pow(input,3); /*Non lineal transconductance*/

    if(x[0]<=lb)    /*Output swing limitation */
        {x[0]=lb;}
    if(x[0]>=ub)
        {x[0]=ub;}

    if(input<=ilb)  /*Input swing limitation */
        {input=ilb;}
    if(input>=iub)
        {input=iub;}

    dx[0]=-go/C*x[0]+gm/C*input+1/C*iin; /* Differential equation to evaluate transient response */
                                         /* 1-pole case
}

static void mdlTerminate(SimStruct *S)
{
}
# ifdef MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cd_sfun.h"
#endif
```

A.2 Código en C del modelo comportamental del integrador Gm-C de la Figura 3.4b incluyendo el efecto de la no linealidad de la ganancia del amplificador [19].

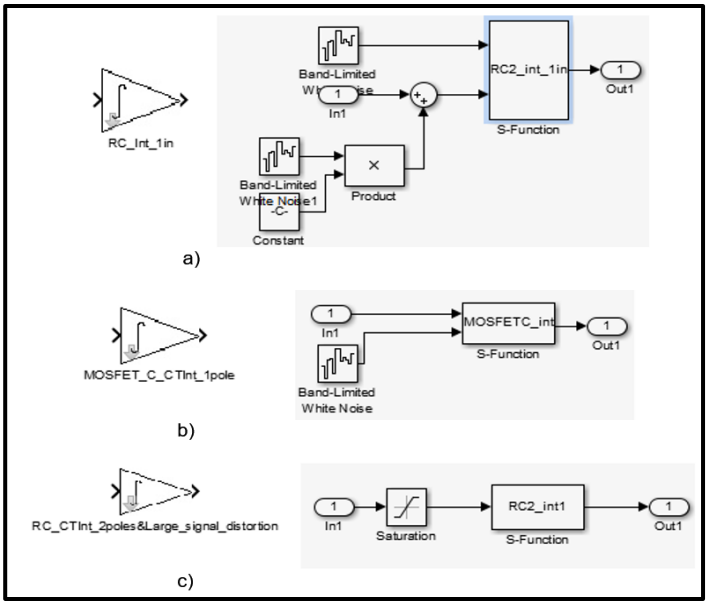
Anexo B

Bloques de integradores y comparadores en SIMSIDES.

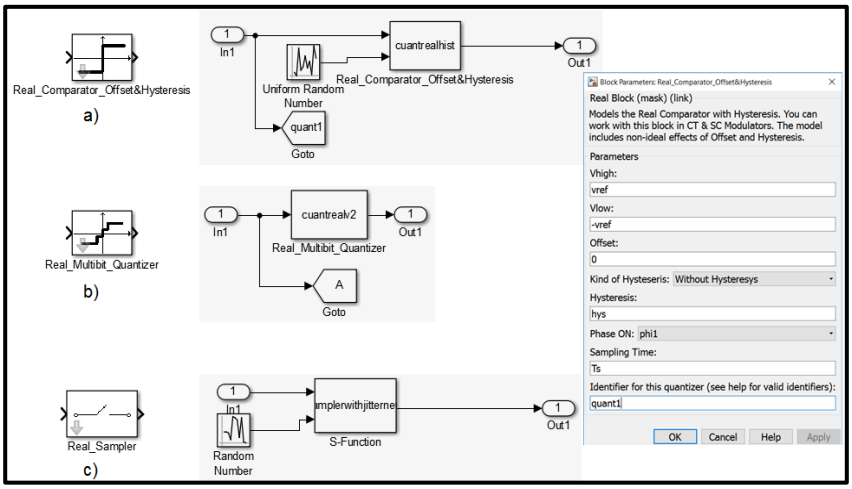
Los integradores pueden ser de tiempo discreto o tiempo continuo (B.1 y B.2). El integrador que se utilizó para realizar los moduladores fue de tiempo discreto (SC_FE_Integrator_All_EffectsII) y un comparador real (B.3a). En cada uno de ellos se contemplan los no ideales.



B.1 Integradores SC FE de tiempo discreto de SIMSIDES en simulink: a) Integrador básico SC_FE, b) Integrador SC_FE con ganancia finita CD y c) integrador básico SC_LD.



B.2 Integradores de tiempo continuo de SIMSIDES en simulink: a) Filtro RC simple, b) Filtro MOSFET, c) Filtro de dos polos RC.

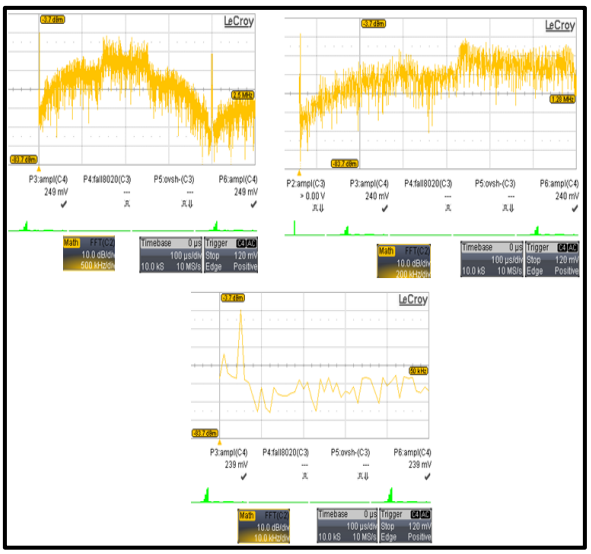


B.3 SIMSIDES: a) Comparador real, b) Cuantizador Multibit real, c) Sampler real.

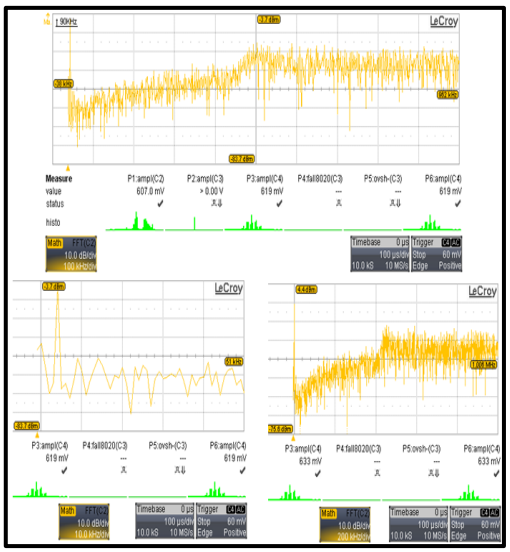
Anexo C

Respuestas de la tarjeta FPAA con diferentes valores de amplitud.

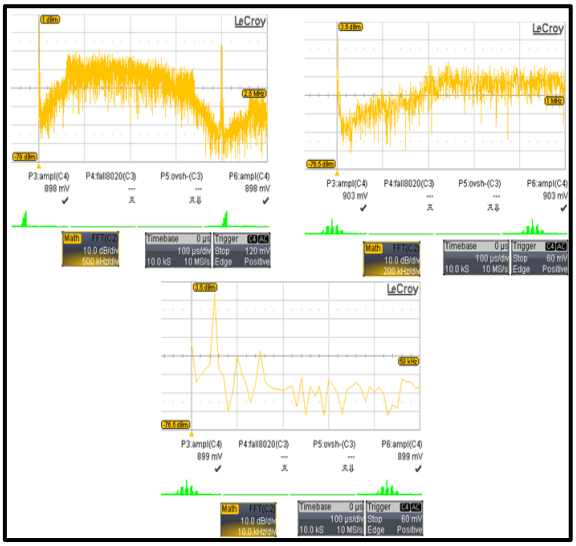
Las respuestas del modulador ΣΔ de segundo arden de la tarjeta AD2 según la variación de la amplitud de entrada de la señal sinusoidal: C.1, donde se aprecia cómo al aumentar la amplitud, la energía se va dirigiendo hacia afuera de la banda de la señal, donde posteriormente será eliminada por un filtro. Se puede ver que de C.1a a C.1b la frecuencia del espectro se traslada hacia la salida. Solo se llegó hasta una amplitud de $1.8v_{pp}$ ya que al rebasar esta amplitud se presentaba una atenuación de la señal.



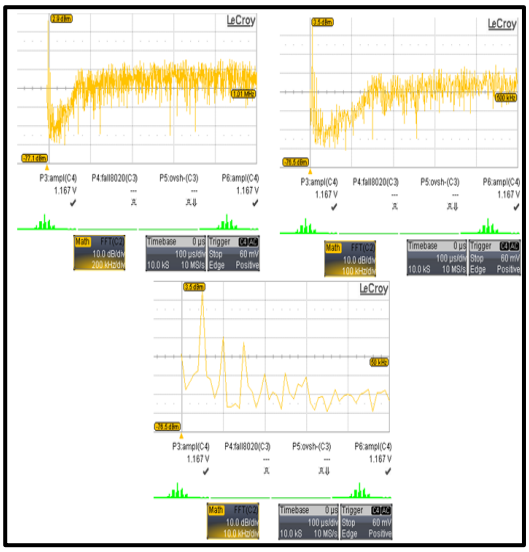
a) Espectro de salida del modulador ΣΔ con Amplitud de $0.3V_{pp}$.



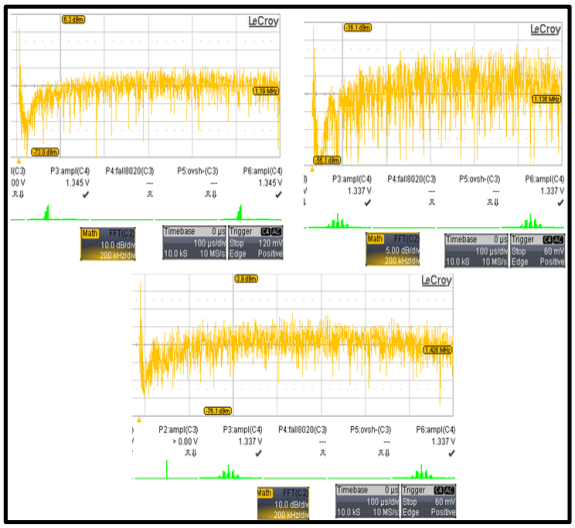
b) Espectro de salida del modulador ΣΔ con Amplitud de $0.7V_{pp}$.



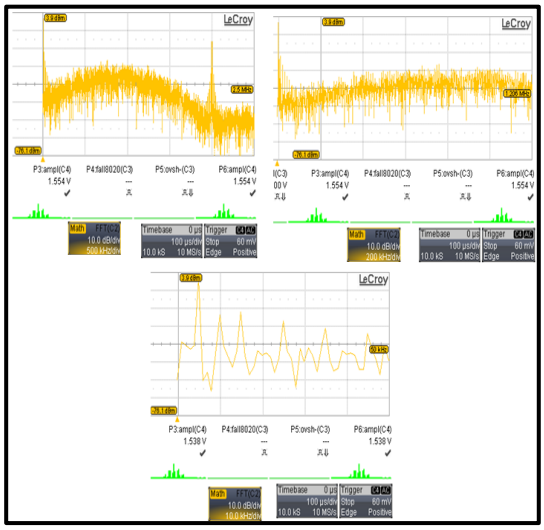
c) Espectro de salida del modulador ΣΔ con Amplitud de $1V_{pp}$.



d) Espectro de salida del modulador ΣΔ con Amplitud de 1.3 V_{pp}.



e) Espectro de salida del modulador ΣΔ con Amplitud de $1.5V_{pp}$.



f) Espectro de salida del modulador ΣΔ con Amplitud de $1.8V_{pp}$

C.1 a)-f): Espectro del modulador ΣΔ de segundo orden de un solo lazo de un solo bit.